ОБ'ЄКТНЕ СЕРЕДОВИЩЕ ДЛЯ ОПТИМІЗАЦІЇ КАЛЕНДАРНИХ ПЛАНІВ ПРОГРАМ

О.В.Сілагін
Вінницький національний технічний університет
Хмельницьке шосе, 95, м. Вінниця, Україна, 21021
E-mail: iti@svitonline.com

Класична задача календарного планування програм має своєю метою створення оптимального календарного графіку виконання деякої множини операцій (робіт) певного проекту [1]. При цьому задача розбивається на декілька етапів :

1) створення мережної моделі у вигляді орієнтованого графа, де дуги графа представляють конкретні операції (роботи), а вершини графа – моменти закінчення одних операцій та початок інших (події).

2) знаходження критичного шляху в графі, що, з урахуванням тривалості операції, відповідає визначенню мінімально можливого часу виконання всього проекту. Критичний шлях визначається за допомогою прямого та зворотного проходів

3) визначення резервів часу для некритичних операцій, таких як вільний та повний резерв.

4) розробка та оптимізація календарного графіку виконання проектів з урахуванням розподілу ресурсів, ймовірнісного та вартісного факторів.

Якщо перші три етапи доволі легко піддаються алгоритмізації, то останні є серйозним каменем спотикання як для математиків, так і для програмістів. Традиційна розробка та оптимізація календарних графіків проводилась на занадто спрощених моделях із застосуванням універсальних або спеціалізованих середовищ моделювання типу MatCAD. Через математичні труднощі дотепер не розроблений метод, що забезпечує знаходження оптимального рішення з використанням хоча б одного з критеріїв (наприклад, мінімізації ресурсів). Тому для прийняття рішень користуються евристичними алгоритмами і з застосуванням експертних оцінок. Як правило, в інтерактивному режимі по найважливішому із критеріїв будується свій, близький для оптимального, календарний графік, який потім корегується з урахуванням інших критеріїв.

Такий підхід має кілька недоліків: по-перше, недостатня адекватність моделі, зв'язана з математичною складністю багатокритеріального моделювання в універсальних середовищах; по-друге, потребує високої математичної та програмістської підготовки та досвіду користувачів, оскільки процес оптимізації здійснюється в інтерактивному режимі з поступовим залученням критеріїв. Це закривало дорогу до використання систем оптимізації та прийняття рішень в календарному плануванні програм широкому загалу інженерно-економічних працівників.

Об'єктна парадигма програмування, а особливо створення спеціалізованих об'єктних середовищ «шаблонів» та «ідіом» типу "hush", "crush" та інших [2], дає можливість розробляти та накопичувати більш адекватні, багатокритеріальні моделі календарних програм. Одночасно це спрощує побудову евристичних алгоритмів оптимізації і відповідно розширює коло користувачів подібних систем прийняття рішень.

На кафедрі інтелектуальних систем Вінницького національного технічного університету розпочата робота по моделюванню календарних графіків планування програм у вигляді шаблонів та ідіом об'єктного середовища «hush». При аналізі доцільності використання середовища «hush» виявлено, що є деяке протиріччя між бажанням побудувати чітку об'єктну модель і бажанням досягнути тієї гнучкості, що необхідна для підтримки підходу з орієнтацією на різні парадигми програмування. Це протиріччя може бути подолане, якщо виділити два рівні: об'єктну модель, що надається середовищем «hush» в розпорядження користувачів середньої кваліфікації та об'єктну модель, що призначена для висококваліфікованих користувачів.

Розробка застосування середовища «hush», як середовища для прийняття рішень в календарному плануванні, містить вирішення кількох програмістських задач, в тому числі програмування на системному рівні (наприклад, мережні функції), програмування інтерфейсу користувача (що включає планування загального вигляду екрану і визначення реакції елементів управління на дії користувача), і опис (на високому рівні) прикладних функцій. Для кожної із цих задач потрібен свій підхід і, можливо, власна прикладна мова програмування. Наприклад, розробка інтерфейсу користувача найбільш зручно виконується за допомогою сценаріїв, що дозволяє мінімізувати втрати часу на компіляцію та зв'язування. Таким же чином, при визначенні специфічних для застосувань функціональних можливостей рівня маніпулювання знаннями, може використовуватись декларативна або логічна мова програмування. В інтегрованому середовищі «hush» з самого початку була закладена підтримка підходу, що допускає розробку програм, які орієнтовані на різні парадигми програмування. Як наслідок середовище визначає також засоби взаємодії між мовами з різними парадигмами, наприклад, між мовою C++ і мовою сценаріїв, подібних Tcl. Сучасні мови сценаріїв такі як Tcl або Python також мають засоби впровадження їх програм в мови C та C++. Але розширення цих мов функціями, що визначені на мовах C та C++, так і використання цих мов в середині програм на C та C++, як і раніше, залишається доволі незручним. Бібліотека «hush» надає єдиний інтерфейс для ряду мов сценаріїв і, крім того, пропонує набір елементів оформлення і мультимедійних розширень, доступних як і з будь-якого інтерпретатора сценаріїв, так і через звичайний інтерфейс мови C++.

Ці концепції втілені в псевдоабстрактних класах, які реалізовані з використанням стандартних для середовища «hush» ідіом. Середовище «hush» надає користувачам такі стандартні класи:

- session - управління додатком і його частинами

- kit - надання доступу до операційної системи та інтерпретаторів
- handler - зв'язування з подіями функцій, написаних на мові C++
- event - збереження інформації, зв'язаної діями користувача, або системними подіями
- widget - відображення інформації на екрані
- item - надання елементів окремих компонент інтерфейсу користувача.

Для створення додатку в середовищі розробки «hush» потрібно визначити його клас, породжений від класу session. Він призначений для ініціалізації додатку і запуску головного циклу віконного інтерфейсу. Крім того, за допомогою функції kit::bind можна зв'язати написані на мовах C++ та Java об'єкти-обробники (handler) з відповідними командами мови сценаріїв. Об'єкти-обробники - це об'єктна реалізація механізму функції зворотного виклику. Перевага цього варіанту реалізації є те, що доступ до клієнтських даних може бути реалізовано способом, безпечним з точки зору типізації (за допомогою ресурсів, пам'ять для яких була виділена при створенні обробника, або за допомогою інформації, що передається через механізм подій). При активізації об'єкт-обробник одержує покажчик на подію. Клас widget (елемент оформлення) та item (графічний елемент) породжені від класу handler, що дозволяє елементам оформлення і графічним елементам бути своїми власними обробниками.

В застосування до задачі календарного планування користувач одержує ідіому grafic, на основі якої створює родину власних екземплярів класів-додатків, породжених від класу item. Одночасно для набуття та маніпулювання експертними знаннями створюються екземпляри класів statistica, expert, resours, як породжені від класу event.

В даний час проводиться формування та удосконалення баз експертних знань для поліграфічних та видавничих проектів, що дозволить провести апробацію задекларованого підходу в даній галузі.

ЛІТЕРАТУРА

1. Таха Х.А. Введение в исследование операций. Ч.2 : Пер. с англ. – М: Мир, 1987г. – 387с.
2. Элиенс А. Принципы объектно-ориентированной разработки программ. 2-е издание. :Пер. с англ. – М: Вильямс, 2002 – 496с.