Е.М. ЛАВРИЩЕВА

УДК 681.3.06

СБОРОЧНОЕ ПРОГРАММИРОВАНИЕ. Теория и практика

Ключевые слова: *метод сборки, интерфейс, типы, структуры данных, отображение данных, технология разработки, автоматизация сборки*.

ВВЕДЕНИЕ

В настоящей работе представлена динамика развития сборочного программирования за последние двадцать лет. Основные его положения сформировались в 1970-1990 гг. как результат разработки системы автоматизации производства программ (АПРОП), инициированной В.М. Глушковым в 1974 г. [1] и выполненной под научным руководством Е.Л. Ющенко. Фундаментальные основы сборочного программирования обсуждались на семинарах и конференциях разного уровня, защищены в докторской диссертации [2] и отражены в ряде публикаций [3–9].

Объекты сборочного программирования — готовые модули (макромодули, подпрограммы, функции, программы и др.), описанные на разных языках программирования (ЯП) четвертого поколения (Алгол, Фортран, ПЛ/1, Кобол и др.). Такие модули (далее — программные объекты) накапливались в исходном и выходном кодах в библиотеках ЭВМ, в фондах алгоритмов и программ (в текстовом виде на бумажном носителе), а также в архивах самих разработчиков.

Метод сборки — способ сопряжения разноязыковых объектов в ЯП, основанный на теории спецификации и отображения (mapping) типов и структур данных ЯП, представленных алгебраической системой [10, 11]. Основу алгебраического формализма составляют типы данных, операции над ними и функции релевантного, эквивалентного преобразования одних типов в другие. Сопряжение пар объектов в ЯП осуществляется через оператор вызова, в списке параметров которого задаются значения формальным параметрам, проверяемым на соответствие типов данных с помощью утверждений алгебраической системы, доказывающих необходимые и достаточные условия отображения данных в классе ЯП. Результат отображения — операторы релевантного преобразования типов данных в модуле-посреднике сопрягаемых объектов.

Одновременно с данным подходом развивались и другие методы, основанные на повторном использовании готовых объектов и их интерфейсов, средствах обеспечения взаимодействия разноязыковых объектов в классе современных ЯП (Паскаль, Бейсик, C#, Java, Smalltalk и др.) и операционных сред [12–19]. Интерфейс объектов развивался в направлении его формального описания (IDL, API и др.), стандартизации, систематизации артефактов повторного использования (компонентов, сервисов, аспектов и др.) и накопления в библиотеках (Matlab, IP, Demral и др.) или репозитариях. Объекты повторного использования, как готовые ресурсы программирования, используются при изготовлении больших программных систем и их семейств [19–24].

1. МЕТОД СБОРКИ ОБЪЕКТОВ. ОТОБРАЖЕНИЕ ТИПОВ ДАННЫХ ЯП

Основная задача метода сборки — сопряжение разноязыковых объектов в ЯП [7–9] путем отображения типов и структур данных из множества $T = (X, \Theta)$ языков $L = \{l_\alpha\}$, где X — множество значений типов данных, Θ — множество операций преобразования отличающихся типов данных в одном языке l_α в типы данных другого l_β. Под отображением типа $T_\alpha^i = (X_\alpha^i, \Theta_\alpha^i)$ в тип $T_\beta^k = (X_\beta^k, \Theta_\beta^k)$ языков l_α, l_β понимается приведение значения переменной X_α^i к значению $X_\beta^k \in X$ операциями Θ_α^j и $\Theta_\beta^k \in \Theta$.

Сущность отображения типов данных состоит в следующем:

— определение операций и функций преобразования типов данных T_α^t, T_β^k сопрягаемых объектов в классе языков L;

— выполнение для каждой пары сопрягаемых разноязыковых объектов на языках l_α и l_β операций отображения и/или селектора S, конструктора C.

В общем случае множество T типов данных языков L включает наборы простых типов данных ($t = b(\text{bool}), c(\text{char}), i(\text{int}), r(\text{real})$) и сложных (структурных) типов данных ($t = a(\text{array}), z(\text{record}), u(\text{union}), e(\text{enum}),\dots$) как комбинаций простых типов данных. Для каждой пары $l_\alpha, l_\beta \in L$ построены алгебраические системы, включающие пары систем отображения простых типов данных Ω_1 и сложных — Ω_2 в классе языков L:

$$\Omega_1 = \{G_\alpha^b, G_\beta^b\}, \{G_\alpha^c, G_\beta^c\}, \{G_\alpha^i, G_\beta^i\}, \{G_\alpha^r, G_\beta^r\},$$

$$\Omega_2 = \{\{G_\alpha^a, G_\beta^a\}, \{G_\alpha^z, G_\beta^z\}, \{G_\alpha^u, G_\beta^u\}, \{G_\alpha^e, G_\beta^e\},\dots\}. \quad (1)$$

В системе Ω_1 каждая пара систем G_α^t, G_β^t обеспечивает отображение простых типов данных на множестве значений и операций: $G_\alpha^t = < X_\alpha^t, \Theta_\alpha^t >, G_\beta^t = < X_\beta^t, \Theta_\beta^t >$, где тип данных $t = b, c, i, r, a, z, u, e$ и Θ_α^t, Θ_β^t — операции отображения пары языков l_α, l_β. Аналогично определяются системы Ω_2 сложных типов данных $t = a, z, u, e$.

1.1. Теория отображения простых типов данных ЯП. Отображение типа данных $t \to q$ систем G_α^t и G_β^q в классе пар языков l_α и l_β обладает следующими свойствами:

1) системы G_α^t и G_β^q изоморфны, если они построены для одинаковых типов данных языков l_α, l_β и множества их значений, т.е. G_α^t и G_β^q совпадают;

2) между значениями типов данных X_α^t и X_β^q существует изоморфизм, если множества их операций Θ_α^t и Θ_β^q различны, $\Theta = \Theta_\alpha^t \cap \Theta_\beta^q$ не пусто и существует изоморфизм двух систем — G_α^t и G_β^q из Ω_1, каждая из которых имеет вид $G_\alpha^t = < X_\alpha^t, \Theta_\alpha^t >$, $G_\beta^q = < X_\beta^q, \Theta_\beta^q >$;

3) для типов строка t и вещественный q не существует изоморфизма между X_α^t и X_β^q;

4) мощности систем $|G_\alpha^t| = |G_\beta^q|$ должны быть равны, что является критерием оценки правильности сопряжения объектов в разных языках L.

Среди множества операций Θ могут быть операции отношения, которые определяют отношение линейного порядка элементов на множестве значений X_α^t. Изо-

морфизм двух систем из Ω_1 доказывается на перечислимых типах l_α и l_β множества L (например, на символьных типах G_α^c и G_β^c).

Теорема 1. Пусть φ — отображение системы $G_\alpha^c \in \Omega_1$ в систему $G_\beta^c \in \Omega_1$.

Отображение φ изоморфно тогда и только тогда, когда φ изоморфно отображает значения типов данных X_α^c в X_β^c с сохранением линейного порядка.

Необходимость. Пусть φ — изоморфизм. Тогда при отображении G_α^c в G_β^c сохраняются все операции множества $\Theta = \Theta_\alpha^c = \Theta_\beta^c$, в том числе и операция отношения, определяющая линейный порядок множества значений X_α^c и X_β^c.

Достаточность. Пусть φ изоморфно отображает X_α^c в X_β^c и сохраняет линейный порядок элементов. Тогда проверка сохранности операций отношения выполняется согласно упорядоченности элементов.

Теорема доказана. Она позволяет доказывать любое изоморфное отображение элементов множеств X_α^c и X_β^c.

Лемма 1. Для любого изоморфного отображения φ между системами G_α^t и G_β^q выполняются равенства

$$\varphi_1(X_{\alpha.\min}^t) = X_{\beta.\min}^q, \quad \varphi_2(X_{\alpha.\max}^t) = X_{\beta.\max}^q. \tag{2}$$

Минимальное и максимальное значения $X_{\alpha.\min}^t$ и $X_{\alpha.\max}^t$ этим элементам множества не принадлежат, а принадлежат соответственно элементам X_α^t и X_β^q. Из этого следует, что $X_{\alpha.\min}^t$ и $X_{\alpha.\max}^t$ входят в алгебраическую систему (1).

Ввиду леммы 1 и равенства минимальных значений $\varphi(X_{\alpha.\min}^c) = X_{\beta.\min}^c$ для $t = c$ доказывается операция succ. Последовательно применяя операцию succ к этому равенству и учитывая линейную упорядоченность X_α^c и X_β^c ($x < \mathrm{succ}(x)$), получаем, что для любого $x_\alpha^c \in X_\alpha^c$ и $x_\alpha^c \neq X_{\alpha.\min}^c$ из равенства $\varphi(X_\alpha^c) = x_\beta^c$, где $x_\beta^c \in X_\beta^c$, следует $\varphi(\mathrm{succ}(x_\alpha^c)) = \mathrm{succ}(x_\beta^c)$.

Аналогично доказывается операция pred с помощью равенства $\varphi(X_{\alpha.\max}^c) = X_{\beta.\max}^c$ и любого $x_\alpha^c \in X_\alpha^c$ и $x_\alpha^c \neq X_{\alpha.\max}^c$. Из равенства $\varphi(X_{\alpha.\max}^c) = x_\beta^c$ следует равенство $\varphi(\mathrm{pred}(x_\alpha^c)) = \mathrm{pred}(x_\beta^c)$.

Теорема 2. Любой изоморфизм φ между системами G_α^b и G_β^b булевых типов — тождественный изоморфизм:

$$\varphi(X_{\alpha.\mathrm{false}}^b) = X_{\beta.\mathrm{false}}^b, \ \varphi(X_{\alpha.\mathrm{true}}^b) = X_{\beta.\mathrm{true}}^b. \tag{3}$$

Доказательство. При отображении G_α^b и G_β^b всегда справедливо $X_{\alpha.\mathrm{false}}^b < X_{\beta.\mathrm{true}}^b$. Учитывая, что φ сохраняет линейный порядок, единственным изоморфизмом является (3).

Теорема 3. Любой изоморфизм между алгебраическими системами с соответствующими числовыми типами — тождественный автоморфизм.

Доказательство этой теоремы проводится аналогично теореме 2.

Отображение символьного типа в целый и целого типа в символьный выполняется с помощью систем G_α^c и G_β^i, G_β^i и G_α^c. Пересечение множеств операций $\Theta' = \Theta_\alpha^c \bigcap \Theta_\beta^i$ включает множество операций отношения $\Theta' = \{\text{pred, succ, } \le\}$, которое в точности совпадает с множеством операций любого перечислимого типа. Можно воспользоваться результатами теоремы 3 и выбрать любое изоморфное отображение множеств вида X_α^c и X_β^i, которое сохраняет линейный порядок. В качестве X_β^i может быть выбран любой отрезок целого типа, для которого $|X_\alpha^c| = X_{\beta.\max}^i - X_{\beta.\min}^i + 1$ и должно выполняться условие о равенстве мощностей основных множеств $|G_\alpha^t| = |G_\beta^q|$.

Отображение символов в целое можно выполнить путем подстановки каждому символу его порядкового номера в алфавите — функция ord, и наоборот, по порядковому номеру символа в алфавите подставить его значение — функция chr.

Отображение целого в булевой тип и булевого в целый определяется на пересечении множества операций $\Theta' = \Theta_\alpha^c \bigcap \Theta_\beta^i$. К нему применимы результаты операций отображений над целым и булевым типами, как над перечислимыми типами.

Операции отображения числовых типов определяются на множестве целых чисел Z, рациональных Q и действительных (вещественных) R. Системы G_α^i и G_β^i, соответствующие целым типам, изоморфны. Операция сложения целых чисел обеспечивается равенством $\varphi(x_\alpha^i + y_\alpha^i) = \varphi(x_\alpha^i) + \varphi(y_\alpha^i)$. Полагая $y_\alpha^i = 0$, получаем $\varphi(x_\alpha^i) = \varphi(x_\alpha^i) + 0$. Так как x_α^1 — произвольное, это равенство выполняется только при $\varphi(0) = 0$. Операция умножения выполняется на равенстве $\varphi(x_\alpha^i * y_\alpha^i) = \varphi(x_\alpha^i) * \varphi(y_\alpha^i)$. Если $y_\alpha^i = 1$, то $\varphi(x_\alpha^i) = \varphi(x_\alpha^i) + \varphi(1)$, где $\varphi(x_\alpha^i)$ — целое число. Если оно не равно 0, то выполняется целочисленное деление, т.е. получаем $\varphi(1) = 1$, где φ — изоморфизм между системами G_α^i и G_β^i.

Изоморфное отображение φ между системами G_α^r и G_β^r соответствует вещественным типам, где X_α^r и X_β^r — подмножества R. Так как $Z \subset R$, результат справедлив для целых чисел, а также для подмножества вещественных чисел. Если $x_\alpha^r \in X$, т.е. $x_\alpha^r = m/n$, где m и n — целые числа, то $m = \varphi(m) = \varphi(n(m/n)) = \varphi(n) * \varphi(m/n) = n * \varphi(m/n)$. Отсюда следует равенство $\varphi(m/n) = m/n$, которое выполняется для любых чисел из Q. Вещественное число x можно с заранее определенной точностью ε представить в виде рационального числа m/n так, что $|x - m/n| < \varepsilon$.

Таким образом, для любых $x \in R$ выполняется $\varphi(x) = x$. Учитывая, что множества Z и R — базовые для основных множеств X_α^i и X_β^r систем G_α^r и G_β^r, теорема 3 доказана.

1.2. Теория отображения сложных типов данных ЯП. Структурные типы данных строятся из простых типов с помощью операций конструирования. Множества значений этих типов конечны и сохраняют линейный порядок. Отображение структурных типов определяется отображением простых типов данных, из которых они строятся.

Пусть G_α^a и G_β^a — системы из Ω_2, соответствующие массиву в классе L.

Теорема 4. Изоморфные отображения φ_i и φ_v множеств индексов и значений элементов массивов сохраняют линейный порядок.

Изоморфизм определяется отображениями $\varphi_i : I^a_\alpha \to X^a_\beta$, $\varphi_v : Y(X^a_\alpha) \to$ $\to Y(X^a_\beta)$, где $Y(X^a_\alpha)$ — множество значений $x_1, \ldots, x_n$ элементов массива в I^a_α; $Y(x^a) \subset X^i$; X^t — множество значений элементов массива типа T, заданное на множестве индексов этого массива.

Изоморфизм между системами G^a_α и G^a_β определяется отображениями φ_i и φ_v, которые сохраняют линейный порядок элементов этого массива.

Отображение записи. Пусть G^z_α и G^z_β — две системы Ω_2, соответствующие типу данных: запись в двух языках — l_α и l_β, $x^z_\alpha \in X^z_\alpha$, $x^z_\beta \in X^z_\beta$.

Теорема 5. Если между последовательностями элементов записей x^z_α и x^z_β существует взаимно однозначное соответствие, то изоморфизм φ между системами G^z_α и G^z_β определяется изоморфным отображением элементов записи.

Условиями изоморфного отображения множеств значений этого типа является количество компонентов записи и соответствующий порядок их следования.

Для отображения записи в массив необходимо, чтобы типы всех компонентов записи были одинаковые, а сама запись была представлена массивом. Множество значений компонентов записи должно соответствовать множеству значений элементов массива Y. Множество индексов строится простым переупорядочением компонентов записи: i-му компоненту записи ставится в соответствие $\varphi(i)$-й элемент во множестве I индексов значений. При отображении записи в массив используется теорема 4.

Другой подход к преобразованию структурных типов — использование операций селектора S и операций конструирования C для изменения уровня структурирования данных.

Операция селектора S для массива определяется как ограничение отображения: $M : I \to Y$ на I', $E \# M : I \to Y$, где E — вложение $I' \in I$. Тогда $M / \{k\}$ соответствует k-му элементу массива при $I' = k$. Аналогично эта операция определяется и для записи $M / \{S^v_m\}$, где M — отображение компонентов записи, S^v_m — выбор соответствующего компонента записи.

Множество операций селектора представим соотношением $S = \bigcup_{\alpha=1}^{n} S_\alpha$, $S_\alpha = \{S^a_\alpha, S^z_\alpha, S^u_\alpha, S^e_\alpha\}$, где S_α — множество операций селектора структурных типов в ЯП l_α; S^a_α — операция селектора массивов в l_α; S^z_α — операция селектора записей в l_α; S^u_α — операция селектора объединения; S^e_α — операция селектора последовательности.

Операция конструирования C структурных типов данных выполняется при условии, что множество объектов, из которых конструируется новый структурный тип, имеет произвольную природу. При конструировании нового структурного типа выполняются операции: упорядочение множества элементов, определение множества индексов и установление взаимно однозначного соответствия между множествами индексов и значений. В общем виде множество операций конструирования имеет вид

$$C = \bigcup_{\alpha=1}^{n} C_\alpha, \quad C_\alpha = \{C^a_\alpha, C^z_\alpha, C^u_\alpha, C^e_\alpha\},$$

где C^a_α, C^z_α, C^u_α, C^e_α — операции конструирования массива, записи, последовательности и других структур данных в классе языков L.

Предложенные операции прямого и обратного отображения простых и сложных типов данных в классе ЯП реализованы в системе АПРОП [23] как механизм сопряжения разноязыковых объектов в этих ЯП и формирования соответствующих им модулей–посредников.

2. ПРАКТИЧЕСКИЕ АСПЕКТЫ РЕАЛИЗАЦИИ СБОРКИ РАЗНОЯЗЫКОВЫХ ОБЪЕКТОВ

Теория типов и структур данных, основной формальный аппарат отображения простых и структурных типов данных в разных ЯП, сформировалась в 70-90-е годы [10]. Данная теория применялась при объединении и комплексировании модулей в разных ЯП через механизм интерфейса. Ее роль постоянно возрастала в практике программирования в связи с появлением новых платформ компьютеров, ЯП и операционных сред, требующих установления соответствия программ на старых и новых компьютерных платформах.

Проблема взаимосвязи разноязыковых программ с использованием интерфейса реализована в ряде описанных далее систем и новых подходов.

2.1. Сборка разноязыковых объектов в системе АПРОП. Основу сборки составляет разработанная библиотека интерфейсных функций [7–9], каждая из которых отображает один из вариантов преобразования несовпадающих типов и структур данных в классе ЯП (Фортран, Алгол, Кобол, ПЛ1, Ассемблер) ОС ЕС [25]. Ключевые концепции реализации системы следующие:

— модульный объект и оператор вызова в ЯП [7, 17, 18];

— системы Ω_1 и Ω_2, операции и функции отображения данных ЯП;

— модуль-посредник каждой пары сопрягаемых объектов в ЯП;

— паспорт модуля (назначение, список параметров и их типов, среда и др.);

— модель управления выбором готовых объектов из Банка модулей (вычислительная математика, АСУ и др.);

— системные операции (сборка, тестирование объектов, интерфейсных модулей–посредников каждой пары сопрягаемых объектов и их агрегатов).

Система АПРОП генерирует модули-посредники для каждой пары сопрягаемых разноязыковых объектов на основе формальных алгебраических систем Ω_1, Ω_2 и межъязыкового и межмодульного интерфейсов [3–9].

Межъязыковой интерфейс обеспечивает установление соответствия типов данных в классе ЯП, а также определение алгоритмов отображения несовпадающих типов и структур данных конкретных ЯП, реализацию набора функций и/или макроопределений для отображения указанных данных.

В задачу межмодульного интерфейса входит генерация интерфейсных модулей-посредников для сопрягаемых объектов с операторами конвертирования передаваемых данных и передач управления (туда и обратно).

Приведем пример применения библиотечных функций преобразования при трансформации (TRAN) матрицы, описанной в модуле A на языке ПЛ1 по столбцам и в модуле B — на Фортране по строкам. Модуль-посредник содержит сгенерированные операторы обращения к функциям транспонирования матрицы по строкам модуля A (точка входа B') в матрицу по столбцам для B и CALL B. После выполнения модуля B — обратные преобразования (A') для модуля A и возврат в него [7].

Модуль *A*	**Модуль-посредник**	**Модуль *B***
PROC OPTION (main);	**SCECT**	**SUBROUTINE** $B(K, M)$;
DCL $K(4,10)$, $M(5,5)$;	B' : SAVEP 2;	**Integer** $K(4,10)$, $M95, 5$;
do $J = 1$ **to** 4; **do** $L = 1$ **to** 10;	PLAS1 (4,10);	**write** (6,1) (($K(J,L)$, $L = 1,10$),
$K(J,L) = (J-1)*10 + L$;	PLAS1 (5,5);	$J = 1,4$);
end; end;	TRAN (4,10), 2, 1, $p = 0$;	**write** (6,1) (($M(J,L)$, $L = 1,5$),
do $J = 1$ **to** 5;	TRAN (5,5), 2, 2, $p = 0$;	$J = 1,5$);
do $L = 1$ **to** 5;	ASFT	**return** A';
$M(J,L) = (J-1)*5 + L$;	**CALL** B;	**Format** (4013);
end; end;	A' : TRAN (4,10), 2, 1, $p = 0$;	**end;**
CALL $B'(K, M)$;	TRAN (5,5), 2, 2, $p = 0$;	
end A;	**return** A;	
	end;	

Система АПРОП была реализована для ЯП ОС ЕС ЭВМ, внедрена в 52 организациях СССР (в Москве, Ленинграде, Минске, Риге, Ереване и др.) и у многочисленных потребителей ЯП ОС ЕС. Она также использована как базовая сборочная система в инструментально-технологических комплексах Прометей и РУЗА производства встроенных бортовых программ Министерства радиопромышленности СССР [20]. По решению этого министерства АПРОП передана в составе комплексов в отраслевой Ереванский НУЦ (1984), которые поставлялись в разные организации данной отрасли до 1991 г., использующие ЕС ЭВМ для автоматизации различных видов деятельности.

2.2. Реализация сопряжения модулей в других системах. В период широкого использования разных ЯП в ОС ЕС многие программисты сталкивались с проблемами обеспечения интерфейса разноязыковых модулей. Подходы к решению задач сборки модулей отражены в отечественных и зарубежных работах.

Отечественные разработки. К ним относятся интегральный подход к индустрии систем А.П. Ершова [12], концепция структурного конструирования систем Е.Л. Ющенко [13], принцип многоязыковости систем модульного программирования Е.А. Жоголева [14], метод комплексирования модулей В.Н. Орлова [15], модель сборки И.Н. Парасюка [16] и многие другие.

Интегральный подход основан на систематизации и использовании огромного запаса модулей и программ как «чистой экономии труда, которая должна дать стране 2/3 годового производства программных средств за счет совершенствования методов и средств сборочного создания программ, подобно сборке во многих отраслях промышленности».

Концепция структурного конструирования программных продуктов реализована в системе Мультипроцессист, в которой программы описывались спецификациями системы алгоритмических алгебр. Компоненты алгебры алгоритмов — схемы программ, клоны, инструменты проектирования и синтеза программ. В настоящее время каркас синтеза для объектных программ и их связей реализован в Rational Rose.

Особенность принципа построения многоязычной системы — сборка модульной структуры из разноязычных модулей на семантическом уровне, в котором накапливались основные понятия (типы данных и их значения) на Фортране и Алголе. Данный уровень — база описания семантики этих ЯП. Все сопряжения разноязыковых модулей описывались в машинно-зависимом языке Автокод, что вносило определенные трудности при реализации управляющей программы для модульной структуры.

Метод комплексирования объектов — идеологический аналог сборки в АПРОП. В нем определены все соглашения о связях и управлении разных модулей на ЯП ОС ЕС, принятые в разных трансляторах, и дано методическое руководство по написанию модулей-переходников для каждой пары ЯП. Рассмотрено шесть пар ЯП: ПЛ1 $\leftrightarrow$ Ассемблер, ПЛ1 $\leftrightarrow$ Фортран, Фортран $\leftrightarrow$ Ассемблер. Приведены конкретные примеры модулей–переходников как образцов для практической сборки.

Модель сборки семейства пакетов прикладных программ (ППП) ДЕЛЬТАСТАТ наполнялась функциональными Фортран-модулями, реализующими задачи из прикладной статистики. Для сборки ППП разработано семейство языков спецификаций моделей предметных областей и языков общения. На их основе автоматически собирались Δ-системы семейства ППП, каждая из которых использовала сборочную модель вычислений программ в динамике выполнения.

Зарубежные работы. Интерфейс разнотипных модулей обеспечивали в проектах MIL, SAA, OBERON (www.oberon.ethz.ch/archives/documentation_new). Особенность этих подходов — создание языков описания интерфейсов сопрягаемых объектов и реализация на их основе задач отображения типов и структур данных, передаваемых через параметры. Они предвосхитили появление в 90-х годах языков спецификации интерфейса IDL (Interface Defiinion Language), связей программных

компонентов APL (Application Program Language) и др. Среди проектов наиболее близка к сборке разноязыковых модулей (на примере Паскаля, Modula-2) система OBERON. Она расширена новыми возможностями по изготовлению программных продуктов в современных операционных средах.

Проблема интерфейса модулей была на повестке дня многих международных и отечественных конференций. ГКНТ СССР провел конкурс «Интерфейс СЭВ» (1987). В нем приняли участие специалисты Института кибернетики АН УССР (группы авторов от АПРОП и РТК [26], в работах которых представлен программный и графический интерфейс соответственно).

2.3. Взаимодействие разноязыковых программ в современных ЯП. Проблеме взаимодействия разноязыковых программ в новых ЯП (Си/C++, Visual C++, Visual Basic, Matlab, Smalltalk, Lava, LabView, Perl) посвящена монография И. Бея [19]. В ней отражена многолетняя и системная работа по созданию разных вариантов (более 60) интерфейсных программ с преобразованием несовпадающих типов данных для каждой пары разноязыковых программ и их взаимодействия в современных средах. Это подтверждает актуальность идеи отображения типов данных в новых ЯП.

Им установлены конкретные виды связей каждой допустимой пары программ в ЯП из указанного множества и разработаны функции отображения данных, передаваемых через операторы вызова программ в разных ЯП. Фактически предложено множество разных видов схем описания разноязыковых программ в ЯП, которыми могут пользоваться программисты. Примеры взаимодействия разноязыковых программ проиллюстрированы в современной наглядной форме (панели, сценарии, иконки и образцы) и могут служить конкретным образцом для программистов.

2.4. Принципы взаимодействия программ в среде MS .Net. Типы данных, общие для всех ЯП среды .Net, представлены системой CTS (Common Type System) и включают: статический тип-значение (value type), динамический, ссылочный тип (reference type), интерфейсный тип и указатель. Данная система обеспечивает преобразование указанных типов к системной форме представления и, наоборот, из нее к описанию в ЯП, устанавливает взаимно однозначное соответствие между простыми типами в *C#* и типами FCL-класса (Framework Class Library). Основное назначение ссылочных типов состоит в обеспечении связей разноязыковых компонентов и их повторном использовании.

2.5. Стандартизация типов данных и новых видов ресурсов. На данное время проведена стандартизация типов данных для всех современных ЯП, а также информационных и программных ресурсов Интернета.

Универсализация типов данных. Согласно ДСТУ 3901-99 (ГОСТ 30664-99), гармонизированному с ISO/IEC 11404:1996, концепция интерфейса ЯП основана на отображении языково-независимых типов данных (LIDT — Language Independent Datatypes) и определила решение проблемы интерфейса для всех существующих и появляющихся ЯП. Средства описания LIDT независимы от ЯП и аналогичны средствам описания интерфейсов IDL, RPC и API. Язык LIDT допускает: внешнее преобразование типов данных ЯП в LIDT; внутреннее преобразование LIDT в типы данных ЯП и обратное внутреннее преобразование.

Внешнее преобразование — отображение типа данных любого ЯП в допустимые LIDT. Внутреннее отображение — преобразование LIDT или их семейства в конкретный внутренний тип данных ЯП (в стандарте приведены примеры преобразования типов данных Паскаля, Фортрана, MUMPS в LIDT). Обратное внутреннее преобразование — отображение типа данных ЯП в LIDT в рамках построенного внутреннего преобразования.

Последняя версия ISO/IEC 11404 «Типы данных общего назначения» появилась в 2007 г. и ориентирована на спецификацию сущностей, обрабатываемых в сетевых сервисах любых компьютерных сетей.

Стандартизация ресурсов. Основные ресурсы повторного использования — компоненты и сервисы, используемые в распределенных средах и веб-сервисах [21–23]. Средство спецификации этих ресурсов — RDF (Resource Definition Framework) и схема RDF для их объединения, а средство интеграции сервисов — Semantic Web, в котором сервисы взаимодействуют между собой через механизм вызова или обращения в стандарте W3C (World Wide Web Consortium). Компоненты и сервисы описываются в стандарте WSDL (Web Services Description Language) в виде набора именованных описаний интерфейсов, связывания и типов данных, изоморфных простым типам данных [27].

Таким образом, на данный момент один из путей развития программирования — стандартизация типов данных и новых видов ресурсов (компонентов, сервисов) и их интерфейсов.

3. РАЗВИТИЕ ПРОБЛЕМАТИКИ СБОРОЧНОГО ПРОГРАММИРОВАНИЯ

Задача сопряжения разноязыковых и разнотипных объектов решается в новом фундаментальном проекте НАН Украины (2007–2011) «Разработка теоретического аппарата генерирующего программирования и интегрированной среды его поддержки». Его цель — разработать основы теории и практики генерирующего программирования для автоматизированного создания программных систем и их семейств из готовых программных продуктов. В теорию генерирующего программирования войдет фундаментальный аппарат определения базовых понятий: модели предметных областей; методы их трансформации; инфраструктура компонентов повторного использования для производства семейств систем; новые методы их тестирования и оценки качества, а также создания конфигураций продуктов семейства и др. [21–24].

Сборка сгенерированных с ЯП и для разных сред компонентов и программ системы семейств является более сложной проблемой. Ее сложность определяют типы данных устаревших и новых ЯП, новые методы (компонентные, аспектные, сервисные и др.) или комбинированные с модульным, объектным методами и средствами разработки отдельных программных систем, разнообразие видов готовых ресурсов (современного и наследованного типа), а также особенности операционных сред (распределенность, безопасность и др.). На сложность также влияет большое разнообразие задач в создаваемом семействе систем, включая бизнесовые, финансовые, экономические, кадровые и т.п.

В данном проекте предполагается расширение алгоритмических систем Ω_1 и Ω_2 новыми типами структур (множествами, списками, указателями и др.), понятийными концептами предметных областей членов семейств и средствами их трансформации новыми методами (компилятивного, уточняющего, оптимизационного видов) [21]. Будут исследованы принципы взаимодействия разноязыковых компонентов с применением новых языков спецификации современных ресурсов (LIDT, RDF, WSDL и др.), методы тестирования, усовершенствованные методы сборки компонентов повторного использования, шаблонов, готовых систем (горизонтального и вертикального типов) и формирования мобильных конфигураций систем семейства.

ЗАКЛЮЧЕНИЕ

Теория и практика сборочного программирования больших программ из более простых, описанных на ЯП четвертого поколения, постоянно развиваются. Появились новые методы (компонентный, сервисный, аспектный, генерационный и др.), обеспечивающие производство программных продуктов из готовых ресурсов. Несомненно, метод сборки разноязыковых и разнотипных программных объектов, разработанный с участием Е.Л. Ющенко, остается методом индустрии программных продуктов на ближайшие десятилетия. Он адаптируется к постоянно изменяющимся операционным средам, учитывает особенности новых ЯП и языков интерфейсов, а также новых видов ресурсов (готовых компонентов, сервисов, систем и др.). Сборочное программирование пополняется новыми типами

и структурами данных современных ЯП, стандартами, теоретическими и практическими концепциями трансформации новых артефактов (моделей, языков спецификаций доменов, семейств систем и др.), определяющих новые подходы к автоматизации изготовления программных продуктов.

СПИСОК ЛИТЕРАТУРЫ

1. Система автоматизации производства программ (АПРОП) / В.М. Глушков, А.А. Стогний, Е.М. Лаврищева и др. — К.: Ин-т кибернетики АН УССР, 1976. — 134 с.
2. Лаврищева Е.М. Модели, методы и средства сборочного программирования в системах обработки данных: Дис. ... д-ра физ.–мат. наук. — Киев: Ин-т кибернетики им. В.М. Глушкова АН УССР, 1989. — 348 с.
3. Грищенко В.Н., Лаврищева Е.М. О создании межъязыкового интерфейса для ОС ЕС // УсиМ. — 1977. — № 1. — С. 34–41.
4. Лаврищева Е.М. Вопросы объединения разноязыковых модулей в ОС ЕС // Программирование. — 1978. — № 1. — С. 22–27.
5. Лаврищева Е.М. Автоматизированное изготовление программных агрегатов из разноязыковых модулей // УсиМ. — 1979. — № 5. — С. 54–60.
6. Лаврищева Е.М. Методика изготовления программных агрегатов // Кибернетика. — 1980. — № 2. — С. 77–82.
7. Лаврищева Е.М., Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС. — М.: Финансы и статистика, 1982. — 127 с.
8. Лаврищева Е.М., Грищенко В.М. Сборочное программирование. — Киев: Наук. думка, 1991. — 213 с.
9. Лаврищева Е.М. Сборочное программирование. Некоторые итоги и перспективы // Проблемы программирования. — 1999. — № 2. — С. 20–32.
10. Агафонов В.Н. Спецификации программ: понятийные средства и их организация. — Новосибирск: Наука, 1987. — 240 с.
11. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. — 3-изд., перераб. и доп. — К.: Наук. думка, 1989. — 376 с.
12. Ершов А.П. Опыт интегрального подхода к актуальной проблематике программного обеспечения // Кибернетика. — 1984. — № 3. — С. 11–21.
13. Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. Многоуровневое структурное конструирование программ. Теоретические основы, инструментарий. — М.: Финансы и статистика, 1989. — 208 с.
14. Жоголев Е.А. Технологические основы модульного программирования // Кибернетика. — 1980. — № 2. — С. 44–49.
15. Орлов В.Н. Комплексирование программ в ОС ЕС. — М.: Финансы и статистика, 1986. — 123 с.
16. Парасюк И.Н., Сергиенко И.В. Инструментально-базовая технология сборочного программирования одного класса интеллектуальных пакетов прикладных программ на ЕС ЭВМ // Тр. междунар. науч.-техн. конф. Секция 3. — Калинин, 1987. — С. 43–47.
17. Лаврищева Е.М. Интерфейс в программировании // Проблеми програмування. — 2007. — № 2. — С. 126–139.
18. Лаврищева Е.М. Становление и развитие модульно-компонентной инженерии программирования в Украине. — Киев, 2008. — 33 с. — (Препр. / НАН Украины. Ин-т кибернетики им. В.М. Глушкова; 2008–1).
19. Бей И. Взаимодействие разноязыковых программ. — М.; С-Петербург; Киев: Вильямс, 2005. — 868 с.
20. Липаев В.В. Технология сборочного программирования. — М.: Радио и связь, 1993. — 272 с.
21. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. — М.; СПб.; Харьков; Минск: Питер, 2005. — 730 с.
22. Грищенко В.Н., Лаврищева Е.М. Методы и средства компонентного программирования // Кибернетика и системный анализ. — 2003. — № 1. — С. 39–55.
23. Лаврищева Е.М. Методы программирования // Теория, инженерия, практика. — Киев: Наук. думка, 2006. — 451 с.
24. Лаврiщева К.М. Генерувальне програмування програмних систем і їх сімейств // Проблеми програмування. — 2009. — № 1. — С. 3–16.
25. Система автоматизации производства программ с режимом мультидоступа / А.Т. Вишня, В.Н. Грищенко, Е.М. Лаврищева, Е.И. Моренцов и др. — Ереван: НУЦ, 1984. — № 93. — 968 с.
26. Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.И. Технологический комплекс производства программ на машинах ЕС ЭВМ. — М.: Статистика, 1980. — 264 с.
27. Web Services Description Language (WSDL). Version 2.0. Part 1: Core Language (http://www.w3.org/TR/2004/WD-wsdl20-).

Поступила 07.07.2009