УДК 004.4'22

А.А. Смагин, А.А. Булаев, С.В. Липатова

МОДЕЛЬ ПОКРЫТИЯ СТРУКТУРЫ ПРОГРАММНОГО КОМПЛЕКСА С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК

Смагин Алексей Аркадьевич, *доктор технических наук, профессор, окончил радиотехнический факультет Ульяновского политехнического института. Заведующий кафедрой «Телекоммуникационные технологии и сети» Ульяновского государственного университета. Имеет статьи, изобретения, монографии в области разработки информационных систем различного назначения. [e-mail: smaginaa1@mail.ru].*

Булаев Алексей Александрович, *окончил факультет математики и информационных технологий УлГУ по специальности «Информационные системы и технологии». Инженер кафедры «Телекоммуникационные технологии и сети» УлГУ. Имеет статьи в области информационных систем и технологий. [e-mail: mail@bulalex.ru].*

Липатова Светлана Валерьевна, *кандидат технических наук, окончила факультет информационных и телекоммуникационных технологий УлГУ. Доцент кафедры «Телекоммуникационные технологии и сети» УлГУ. Имеет статьи в области разработки экспертных и автоматизированных информационных систем. [e-mail: dassegel@mail.ru].*

Аннотация

В настоящей статье предлагается двухкомпонентная модель для проектирования программных комплексов с использованием свободно распространяемых ресурсов (библиотек). Первая модель – функциональная модель – используется для формирования покрытий заданной функциональности программного комплекса, включает в себя матрицу соответствия доступных для использования функций и реализующих их библиотек, а вторая – геометрическая модель – для оценки затрат на его разработку в виде круговой диаграммы.

Представлены алгоритмы определения минимального покрытия и отображения результатов по получаемым затратам на круговой диаграмме в виде площадей секторов, занимаемых библиотеками и оставшимся пространством круга, характеризующим собственные затраты на проектирование.

Для поиска минимального числа библиотек с максимальным количеством покрываемых функций предложено использовать дерево библиотек, которое обеспечивает выбор оптимальных путей по критериям затрат на собственные разработки и затрат на разработку средств по подключению и взаимодействию библиотек с программным комплексом (по количеству функциональных операторов).

Ключевые слова: библиотека, геометрическая модель, дерево библиотек, комплексная модель, критерий затрат, круговая диаграмма, матрица соответствия, проектирование, пространство состояний, функциональная модель, функциональное покрытие, функциональность.

Abstract

This article considers a two-component model for the design of software complexes using freely distributed resources (libraries). The first model, a functional model, is used to form complex tasks, including for their libraries, and the second model, a geometric model, is used for estimating the costs of its development in the form of a pie chart.

The algorithms for determining the minimum coverage and displaying results for the costs obtained on a pie chart in the form of areas of sectors occupied by libraries and the remaining space of the circle, characterizing their own design costs are presented.

In order to search for the minimum number of libraries with the maximum number of covered functions, it is suggested to use the library tree, which provides the selection of optimal paths by the criteria for costs for own development and the costs of developing tools for connecting and interacting libraries with the software complex (by the number of functional operators).

Key words: library, geometric model, library tree, complex model, cost criterion, pie chart, matching matrix, design, state space, functional model, functional coverage, functionality.

Введение

В практике проектирования информационных систем широко применяются комплексные модели, включающие в свой состав несколько модельных компонентов, которые имеют собственные области представления и преобразования данных, отображающие процесс разработки проектируемой системы с разных сторон. Такие модели отличает комплексность, особенностью которой является их взаимодействие между собой по определенным правилам в процессе разработки [1, 2].

Например, часто требуется решение связанных между собой задач: одной – с затратами на проектирование, а второй – с обеспечением заданной функциональности, при этом представление затрат обычно отождествляется с размером площадей геометрических объектов, выражаемых в определенных единицах измерения и пропорциях, а функциональность – с обеспечением полноты реализуемых функций.

Представление в виде геометрических объектов позволяет варьировать формой и размером площадей, а функциональную модель удобно представлять в виде матриц, которые имеют широкий диапазон операций по их преобразованию.

Комплексность моделирования позволяет работать попеременно в двух средах: геометрической и функциональной, и результаты, полученные с помощью геометрического моделирования проектных решений, используются при функциональном моделировании и обратно.

Целью использования комплексной (двухкомпонентной) модели является получение эффективных решений, приводящих к полному покрытию заданной функциональности программного комплекса свободно распространяемыми ресурсами (библиотеками) и собственными разработками, минимизирующими затраты на его разработку.

Комплексная модель покрытия функциональности программного комплекса

Комплексная модель (M) покрытия функциональности программного комплекса в общем виде представлена на рисунке 1 и состоит из функциональной модели (Φ) и геометрической модели (G):

$$M=\Phi\cup G. \quad (1)$$

К геометрической модели относятся: совокупность форм $\{h\}$, метрические характеристики $\{g\}$, которые определяют размеры форм, параметры $\{r\}$, задающие местоположение форм в соответствующем геометрическом пространстве.

Представим геометрическую информацию как:

$$G=(\{h\},\{g\},\{r\}). \quad (2)$$

Функциональная модель представляет собой множество функций, доступных для реализации (F), множество библиотек (L), которые покрывают эти функции, и матрицу их соответствия (C):

$$\Phi=(F,L,C). \quad (3)$$

Пересечением (θ) геометрической и функциональной моделей является множество коэффициентов (K_i), определяющих собственные затраты на программную реализацию i-й функции:

$$G \cap \Phi = \theta. \quad (4)$$

Такой процесс взаимодействия двух моделей обеспечивает контроль его состояния и позволяет вносить корректировки на разных этапах проекта.

Функциональная модель программного комплекса

Программный комплекс может быть построен как полностью собственная разработка, которая включает в себя участие разработчиков разной квалификации с использованием определенных языков программирования, платформ, информационных технологий, а также на основе свободно распространяемых ресурсов – библиотек.

Построение такого комплекса представляет собой разработку отдельных или связанных между собой подсистем, реализующих функции, которые образуют функциональное пространство программного комплекса (в дальнейшем – функциональность). Под понятием функциональности программного комплекса понимается его способность выполнять набор функций, удовлетворяющих заданным потребностям пользователей, а под покрытием функциональности – подбор известных решений с возможной их модернизацией.

Для получения функционального покрытия необходимо предварительно составить список всех функций, входящих в функциональность программного комплекса, и одноимённый список функций, реализуемых свободно распространяемыми библиотеками.

Для получения функционального покрытия заданной функциональности предлагается использовать следующий подход:

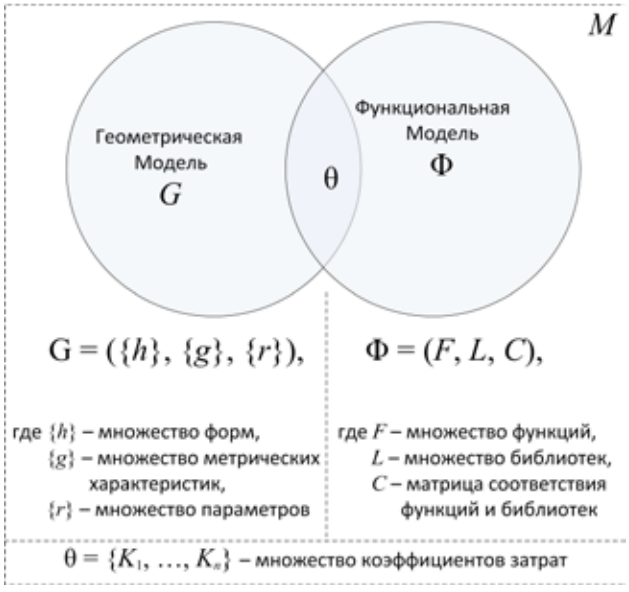

Рис. 1. Комплексная модель покрытия функциональности программного комплекса

1. Строится обобщённая бинарная матрица соответствия (C) функций, которые можно реализовать в рамках заданной области применения, и библиотек, которые частично их покрывают.

$$C = \begin{array}{c|cccc} & f_1 & f_2 & \cdots & f_n \\ l_1 & c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ l_2 & c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_m & c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{array}$$

В этой матрице строки представляют собой множество библиотек $L = \{l_1, \ldots, l_m\}$, а столбцы – множество функций $F = \{f_1, f_2, \ldots, f_n\}$ заданной области применения. В ячейку $c_{i,j}$ обобщённой матрицы соответствия библиотеки l_i и функции f_j записывается единица, если библиотека l_i реализует функцию f_j, в противном случае – ноль.

2. Формируется бинарный вектор $\bar{f}$ функций заданной области применения, каждый элемент которого принимает значение 1, если он принадлежит заданной функциональности программного комплекса, иначе – 0.

3. С целью отсеивания из множества функций тех, которые не входят в заданную функциональность, используется операция умножения матрицы (C) на вектор функций $\bar{f}$.

В полученной в результате перемножения матрице D определяются столбцы, содержащие все нули, после чего эти столбцы удаляются.

Задача поиска библиотек, реализующих заданную функциональность, сводится к нахождению кратчайшего покрытия полученной на предыдущем шаге булевой матрицы D, то есть нахождения такой минимальной совокупности строк матрицы, которая содержала бы не менее одной единицы в каждом столбце матрицы.

Под покрытием матрицы D размера $m \times n$ в данной работе понимается такое подмножество ее строк $i_1, \ldots, i_k$, что для каждого j, где $j = 1, \ldots, n$, найдется такой номер $s = s(j)$, $1 \le s(j) \le k$, что $d_{s(j),\,j} = 1$. Другими словами, подмножество строк $i_1, \ldots, i_k$ матрицы D является ее покрытием, если в подматрице, образованной этими строками, нет нулевых столбцов.

Представляет интерес поиск кратчайшего покрытия матрицы, под которым понимается покрытие матрицы наименьшей мощности (наименьшее количество строк).

Для нахождения покрытий бинарной матрицы можно использовать следующие методы: метод Патрика [3], метод Закревского строчного и столбцового покрытий [4] и другие.

Одним из эффективных методов получения покрытий, широко применяемым на практике, является метод Патрика [3], который обеспечивает получение полного списка покрытий заданной матрицы.

Для нахождения всех функциональных покрытий матрицы D методом Патрика необходимо представить каждый столбец матрицы в виде дизъюнкции библиотек, которые реализуют данную функцию [5].

Рассмотрим пример.

Пусть имеется 5 библиотек и 4 функции, которые необходимо реализовать с помощью этих библиотек.

Матрица D выглядит следующим образом:

$$D=\begin{bmatrix}0&0&1&1\\1&1&0&1\\0&0&1&0\\1&1&0&0\\0&0&0&1\end{bmatrix}\begin{matrix}а\\б\\в\\г\\д\end{matrix}.$$

Первый столбец покрывается строками *б* и *г*, второй столбец – *б* и *г*, третий столбец – *а* и *в*, четвертый столбец – *а*, *б*, *д*.

Составим конъюнкцию перечисленных дизъюнкций:

$$(б\cup г)\cap(б\cup г)\cap(а\cup в)\cap(а\cup б\cup д)=$$
$$=ба\cap бад\cap бва\cap бвд\cap бга\cap бгад\cap бгва\cap$$
$$бгв\cap бгвд\cap га\cap гад\cap гва. \quad (5)$$

Из уравнения следует, что для реализации программного комплекса формируется 12 проектных решений по числу комбинаций библиотек, покрывающих все столбцы матрицы D. Среди них наиболее эффективными с точки зрения количества библиотек будут комбинации библиотек $\{а,б\}$ и $\{а,г\}$.

Геометрическая модель программного комплекса

При всяком проектировании важнейшую роль играет минимизация затрат на разработку, поэтому подбирается некоторое множество критериев, выполнение которых ее обеспечивает. Однако многокритериальность затрудняет во многих случаях процесс анализа их достижимости, и тогда прибегают к выделению основного критерия, а остальные являются дополнительными и служат для исключения неопределенности выбора решения, когда знания значения основного критерия является недостаточными.

Представим геометрическую информацию как:

- форма круга, метрическими характеристиками которого являются: длина радиуса R, площадь S;
- форма сектора круга, метрическими характеристиками которого являются: длина дуги L, угол α', площадь S';
- форма треугольника, метрической характеристикой которого является длина дуги L, угол α'', площадь S'';
- параметры: угол, размеры и местоположение сектора относительно начала системы отсчёта.

Пусть в двумерном пространстве имеется круг радиусом R и покрывающие его геометрические объекты (секторы и составляющие их сегменты и треугольники), где M – общее количество заданных покрывающих объектов (секторов круга с заданными метрическими характеристиками и их теоретико-множественными комбинациями).

Требуется расположить геометрические объекты (секторы) на покрываемой площади круга таким образом, чтобы вся площадь была покрыта целиком.

В качестве геометрической модели выбрана круговая диаграмма (рис. 2), представляющая собой круг с дискретизацией n-секторами одинакового размера, число которых равно $(2*\pi)/n$, где n – количество функций, требуемых для покрытия заданной функциональности проектируемого программного комплекса.

Основным элементом круговой диаграммы выступает геометрическая фигура – единичный сектор, которым представляется функция из заданной функциональности программного комплекса.

Объединением секторов обозначается библиотека или собственная разработка, реализующая набор функций из заданной функциональности, а треугольником соответствующего этой библиотеки сектора круга меньшего радиуса (R_1) – затраты на собственные доработки средств подключения и взаимодействия библиотек.

Библиотека представляет собой информационный объект, реализующий одну или несколько функций, которые могут входить в состав заданной функциональности. Отсюда библиотека, реализующая m-функций, будет отображаться в виде m-единичных секторов.

Единичные секторы, отображающие библиотеку с m-функциями, занимают соседние позиции круговой диаграммы и имеют определённую площадь. У библиотек с разным количеством функций площади соответствующих им объектов круговой диаграммы будут различными.

Две библиотеки называются пересекающимися,

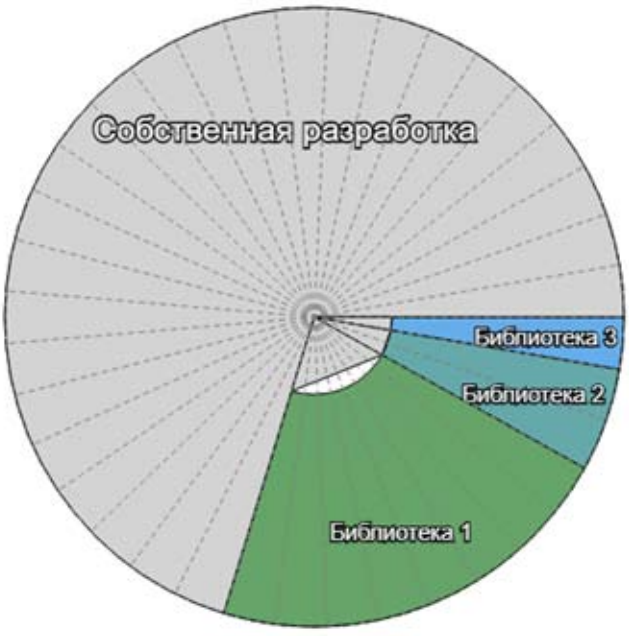

Рис. 2. Круговая диаграмма программного комплекса

если у них имеется хотя бы одна одинаковая функция. Заполнение исходной функциональности программного комплекса может производиться с помощью пересекающихся библиотек, когда одна часть необходимых функций содержится в одной библиотеке, а другая – во второй. На круговой диаграмме обе библиотеки занимают два соседних сектора и отображаются суммарной площадью S_1+S_2. Но на практике одинаковые функции должны быть реализованы только одной из этих библиотек. Поэтому угол сектора библиотеки, которая не реализует одну из повторяющихся функций, на круговой диаграмме должен быть уменьшен. В противном случае при большом количестве пересекающихся библиотек геометрическое покрытие будет избыточным, что приведёт к наложению секторов друг на друга.

Полное покрытие функциональности программного комплекса характеризуется количеством затрат (Z_{API}) на разработку средств подключения и взаимодействия библиотек, выражаемых площадями треугольников соответствующих им секторов круга меньшего радиуса, и затрат $(Z_{соб.раз.})$ на собственные разработки непокрытых заданных функций:

$$Z = Z_{API} + Z_{соб.раз.}. \quad (6)$$

При выборе библиотек встречаются случаи, когда требуется из нескольких библиотек, покрывающих одинаковые функции, выбрать лучшую по некоторому критерию. Таким критерием может служить коэффициент затрат на разработку средств подключения и взаимодействия библиотеки с программным комплексом.

Экспериментальным путём получена зависимость коэффициента затрат на подключение библиотеки от количества функций, которые она покрывает. Данная зависимость вычисляется как отношение площади треугольника затрат библиотеки (S_2), покрывающей n функций, к сумме площадей треугольников (S_1) затрат на подключение каждой из этих функций отдельно:

$$K(n) = \frac{S_2}{n * S_1} = \frac{\sin(n*\varphi)}{n*\sin(\varphi)}, \quad (7)$$

где n – количество функций, покрываемых библиотекой, φ – угол единичного сектора.

На графике (рис. 3) представлена зависимость коэффициента затрат на создание средств подключения и взаимодействия библиотеки с программным комплексом от количества функций, покрываемых этой библиотекой. На основе расчётов выявлено, что с увеличением количества функций, покрываемых библиотекой, уменьшается коэффициент затрат на подключение этой библиотеки.

Рабочий диапазон коэффициента затрат: $[1, N/2)$, где N – суммарное количество функций из заданной функциональности. В случае, если библиотека покрывает одну функцию, то коэффициент затрат на разработку средств подключения и взаимодействия библиотеки с программным комплексом равен 1. При покрытии библиотекой $N/2$ функций коэффициент затрат стремится к нулю.

Сложность собственных разработок покрытия и обеспечения подключения библиотеки к программному комплексу имеет важное значение в процессе его проектирования с учётом использования свободно распространяемых библиотек. Реализация каждой функции, полученной из библиотеки или собственной разработки, имеет свою сложность разработки.

На практике наиболее часто используются количественные метрики программного обеспечения, такие как: количество строк кода (SLOC), цикломатическая сложность, количество операторов, анализ функциональных точек, среднее число строк для функций (классов, файлов, модулей) [3].

В настоящей работе предлагается оценка сложности функции по количеству реализующих её операторов как наиболее удобная для практических применений. В случае, если i-я функция покрывается какой-либо библиотекой, то сложность функции (Z_i) определяется количеством операторов в составе функции этой библиотеки.

Тогда сложность библиотеки будет вычисляться как сумма сложностей функций, которые она покрывает:

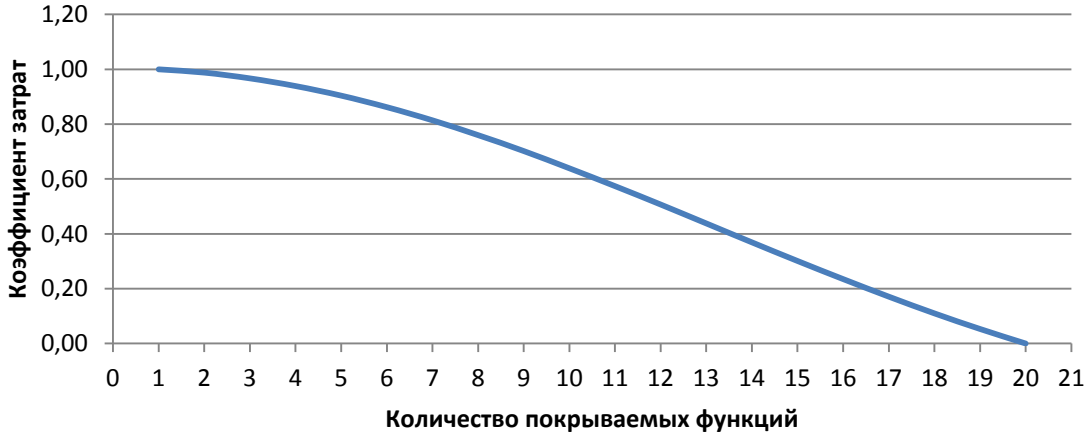

Рис. 3. Зависимость коэффициента затрат на разработку средств подключения и взаимодействия библиотеки с программным комплексом от количества покрываемых функций

$$Z' = \sum_{i=1}^{k} Z_i, \quad (8)$$

где k – количество функций, покрываемых библиотекой.

Зная сложность функций, реализуемых библиотеками, и суммарный угол, который они занимают, можно определить среднее значение сложности на единицу угла, отождествляемое с единицей площади на геометрической модели для измерения затрат на собственные разработки:

$$Z_{ср.} = \frac{\alpha * \sum_{i=1}^{n} Z_i}{360^\circ} \text{ операторов}, \quad (9)$$

где α – угол сектора, занимаемого библиотеками,

n – количество функций, покрываемых библиотеками.

Сложность собственных разработок функций, которые не покрываются библиотеками, предлагается оценивать как произведение угла (β) сектора, который они занимают, на среднее значение сложности $(Z_{ср})$:

$$Z_{соб.раз.} = \beta * Z_{ср.} \text{ операторов}. \quad (10)$$

Взаимодействие моделей при формировании дерева библиотек

Для наглядного представления функциональных покрытий наиболее удобна их визуализация в виде пространства состояний – дерева библиотек с возможностью поиска по нему оптимального покрытия по заданному критерию. Глубина «ветки» дерева библиотек (без начальной вершины) соответствует количеству библиотек в покрытии. Количество «листьев» в дереве библиотек соответствует количеству допустимых покрытий для текущего набора функций [6].

В пространстве состояний, отображаемом в виде дерева библиотек, встречаются библиотеки с различным количеством покрываемых функций. Их выбор влияет на длину пути в дереве и на количество собственных затрат.

Отсюда возникают два варианта построения дерева:

I. Первый вариант: построение дерева по минимальному количеству библиотек. Поиск в дереве идёт сверху вниз, отталкиваясь от корневого узла, а листьями являются множество библиотек, покрывающих заданную функциональность.

Алгоритм построения дерева библиотек имеет вид:

1) Обозначается корневая вершина.

2) Преобразовывается матрица D, исключаются из неё все столбцы, описывающие функции, не входящие в вектор $\bar{f}$.

3) Проводится поиск множества функций $\{f_i\}$, которые реализованы меньшим количеством библиотек, т. е. столбцы с минимальной суммой элементов в преобразованной матрице D:

$$\sum_{j}^{m} c_{i,j} \to min. \quad (11)$$

4) Для каждой функции из полученного множества формируется множество библиотек $\{l_j\}$, реализующих найденные функции в пункте 3.

5) Для каждой из них строится вершина в дереве библиотек и преобразовывается матрица D, из которой исключаются все столбцы, описывающие функции, входящие в библиотеку l_j.

6) Рекурсивно осуществляется переход на шаг 3 для каждой из выбранных библиотек, если в матрице D остались столбцы, в противном случае – строится конечная вершина дерева.

II. Второй вариант: построение дерева по коэффициенту затрат на разработку средств по подключению и взаимодействию библиотек с программным комплексом. В основу алгоритма заложен итерационно-рекурсивный процесс, в котором рекурсия используется для поиска минимального коэффициента затрат в матрице затрат W, а итерация обеспечивает пошаговый подбор библиотеки с минимальными затратами и последующее уменьшение размеров матрицы. Поиск оптимального пути в этом дереве начинается от корня, листьями являются библиотеки с характеризующими их коэффициентами собственных затрат на разработку средств подключения и взаимодействия библиотек с программным комплексом.

Алгоритм построения дерева библиотек с использованием коэффициента затрат:

1) Формируется матрица (W) коэффициентов затрат, в которой столбцами обозначаются функции из заданной функциональности, а строками – библиотеки, их покрывающие. На пересечении строки и столбца матрицы записывается 0, если библиотека не покрывает данную функцию, в противном случае – коэффициент затрат (K_j) на разработку средств подключения и взаимодействия j-й библиотеки с программным комплексом.

Матрица затрат выглядит следующим образом:

$$W = \begin{array}{c|cccc} & f_1 & f_2 & \dots & f_n \\ l_1 & w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ l_2 & w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ l_m & w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{array}$$

где $w_{i,j} = K_j$, если библиотека l_j покрывает функцию f_i, в противном случае – 0.

2) Задаётся корневая вершина дерева.

3) Выполняется поиск ячейки с минимальным коэффициентом затрат на разработку средств подключения и взаимодействия библиотеки с программным комплексом.

4) К корневой вершине добавляется библиотека, соответствующая выбранной на шаге 3 ячейке матрицы W. Если найдено несколько ячеек с одинаковым коэффициентом затрат, то записываются все библиотеки, им соответствующие.

5) Рекурсивно для каждой выбранной библиотеки выполняется удаление из матрицы W строки, соответствующей этой библиотеке, и всех столбцов, которые она покрывает, библиотека указывается в качестве корневой вершины текущего поддерева, и, если матрица W непустая, повторяются шаги 3–5.

Использование второго дерева актуально при создании программных комплексов с ограниченными затратами на разработку, которые можно контролировать с помощью геометрической модели по площади секторов, отражающих собственные разработки и связанных с затратами на программную реализацию.

В случае подбора библиотек может оказаться, что две библиотеки являются эквивалентными, т. е. покрывают одни и те же функции. Для выбора одной из них предлагается делать оценку по дополнительным критериям:

- однородность языков программирования у библиотек;
- платформа, на которой библиотеки функционируют;
- системные и аппаратные требования библиотек;
- критерий неизбыточности кода покрытия библиотек (минимум мощности разности множеств функций из библиотек покрытия и требуемых функций):

$$\left(\sum_{i}^{l}\sum_{j}^{n}\left(\left|c, c_{i,j}=1\right|+\left|\overline{f}\right|-2\left|c, c_{i,j}=1\cap\overline{f}\right|\right)\right)\rightarrow min, \quad (12)$$

где l – количество библиотек в покрытии;

- критерий максимальной расширяемости из l библиотек (максимум мощности множества объединения всех функций библиотек покрытия):

$$\left|\overline{f_1}\cup..\cup\overline{f_l}\right|\rightarrow max, \quad (13)$$

где l – количество библиотек в покрытии.

Поиск оптимального пути на дереве библиотек возможен следующими способами:

- в глубину с учетом выбранного критерия;
- в ширину с учетом выбранного критерия;
- эвристический (эвристика определяется разработчиком).

На практике с большим количеством библиотек возможно использование дополнительных эвристик, которые позволяют сократить вычислительные затраты и не строить всё дерево библиотек. Например, в качестве эвристики можно использовать максимум реализованных функций в библиотеке (для критерия расширяемости). Тогда в алгоритме построения дерева библиотек будут формироваться вершины не для всех библиотек, а только для тех, у которых сумма элементов в строке матрицы D максимально:

$$\sum_{i}^{n} c_{i,j}\rightarrow max. \quad (14)$$

Использование дополнительных критериев и дерева библиотек обеспечивает разработчику гибкость управления процессом проектирования.

В целом, схема взаимодействия функциональной и геометрической моделей в процессе получения проектного решения отображается на рисунке 4, которая показывает, что входными данными в функциональную модель являются функциональность программного комплекса и база

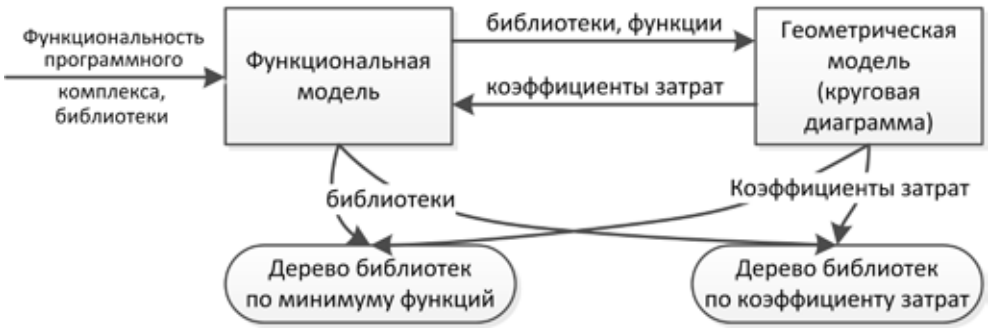

Рис. 4. Схема взаимодействия функциональной и геометрической моделей

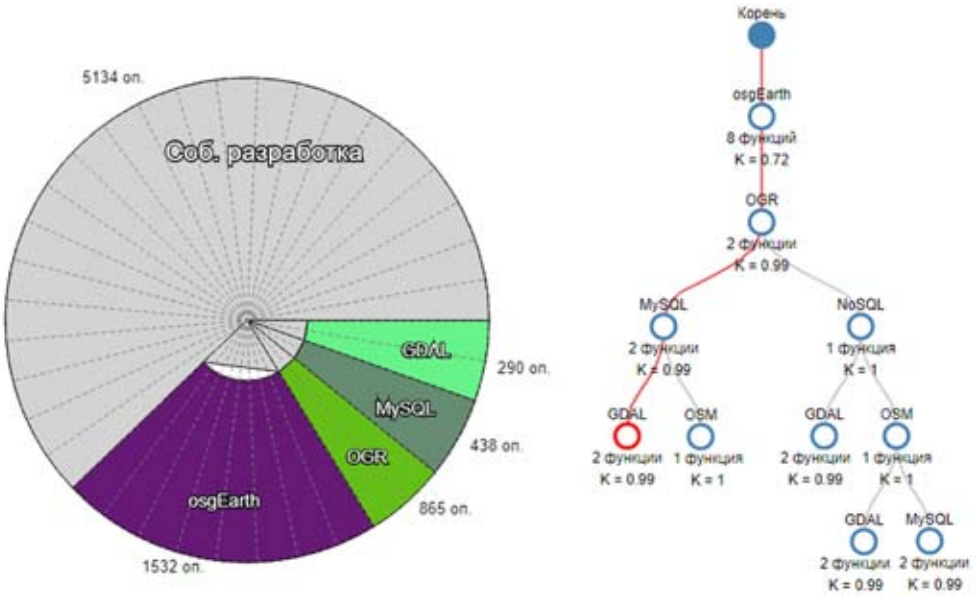

Рис. 5. Дерево библиотек и круговая диаграмма по первому алгоритму

библиотек, которые полностью или частично её покрывают. Между функциональной и геометрической моделями организовано двухстороннее взаимодействие, где из функциональной модели в геометрическую передаётся информация об используемых библиотеках и функциях в виде матрицы соответствия D, а в обратную сторону – вычисленные с помощью геометрической модели коэффициенты затрат на разработку средств по подключению и взаимодействию библиотек с программным комплексом. С использованием функциональной и геометрической моделей имеется возможность построения дерева библиотек двух вариантов: по минимуму функций и по минимуму коэффициентов затрат на подключение библиотек.

Процесс формирования дерева библиотек с помощью функциональной и геометрической моделей по первому алгоритму представлен на рисунке 5. Построение дерева начинается с корня с обращением к матрице D, и последовательном обходе узлов. На круговой диаграмме, расположенной слева, проход каждого узла отображается добавлением сектора с указанием имени библиотеки и количества реализуемых функций. В центре круга выделены площади треугольников, охватывающих секторы соответствующих библиотек и обозначающих необходимость дополнительных затрат (K) по подключению и взаимодействию библиотек с программным комплексом.

Заключение

В настоящей статье предложена модель проектирования программных комплексов с заданной функциональностью, которая обеспечивает эффективность проектирования за счёт экономичного использования свободно распространяемых ресурсов и собственных разработок при программной реализации. Модель состоит из двух компонентов, одна из которых служит для получения минимального покрытия (минимального количества библиотек с максимальным количеством функций) – функциональная модель, а вторая – геометрическая модель, реализованная в виде круговой диаграммы и позволяющая по критериям оценки затрат на собственные разработки и затрат на разработку средств по подключению и взаимодействию библиотек с программным комплексом производить выбор проектных решений. Результаты моделирования отображаются в виде дерева покрытий с двумя алгоритмами его построения по разным критериям, что обеспечивает многовариантность формирования проектных решений программного комплекса.

Предложенный подход и комплексная модель были апробированы в процессе разработки трехмерной геоинформационной системы (3D-ГИС) отображения ситуационной обстановки [7, 8]. Функциональная модель отображала множество функций по трёхмерной визуализации поверхности Земли, 3D-объектов обстановки в заданном районе Земли, текстур, высот и глубин местности, имитации движения объектов и взаимодействию 3D-ГИС с внешним миром. Использование геометрической модели позволило оценить масштабы примерных затрат на создание 3D-ГИС и сформировать её структуру с использованием свободно распространяемых библиотек и собственных разработок [9, 10].

СПИСОК ЛИТЕРАТУРЫ

1. Шелухин О.И. Моделирование информационных систем : учеб. пособие для вузов. – 2-е изд., перераб. и доп. – М. : Горячая линия-Телеком, 2012. – 516 с.

2. Васильев К.К., Павлыгин Э.Д., Гуторов А.С. Многомодельные алгоритмы обработки данных системы мобильных РЛС // Автоматизация процессов управления. – 2014. – № 4 (38). – С. 4–13.

3. Дискретная математика и математические вопросы кибернетики / Ю.Л. Васильев, Ф.Я. Ветухновский, В.В. Глаголев, Ю.И. Журавлев, В.И. Левенштейн, С.В. Яблонский. – М. : Наука, 1974.

4. Закревский А.Д., Торопов Н.Р. Минимизация булевых функций многих переменных в классе ДНФ – итеративный метод и программная реализация // ПДМ. – 2009. – № 1 (3). – С. 5–14.

5. Андерсон Дж. Дискретная математика и комбинаторика. – М. : Вильямс, 2006. – 960 с.

6. Эддоус М., Стэнсфилд Р. Методы принятия решений / пер. с англ. под ред. член-корр. РАН И.И. Елисеевой. – М. : Аудит, ЮНИТИ, 1997. – 590 с.

7. Андрианов Д.Е. Геоинформационные системы: исследование, анализ и разработка. – М. : Государственный научный центр Российской Федерации – ВНИИгеосистем, 2004. – 184 с.

8. Дышленко С.Г., Цветков В.Я. Особенности проектирования ГИС пользователя на основе базового комплекта ГИС «Карта 2011» // Землеустройство, кадастр и мониторинг земель . – 2010. – № 8. – С. 79–84.

9. CASE-средство проектирования 3D-ГИС на основе свободно распространяемых библиотек / А.А. Булаев, С.В. Липатова, Д.А. Мерзляков, А.А. Смагин // Автоматизация процессов управления. – 2016. – № 2 (44). – С. 35–44.

10. Булаев А.А., Липатова С.В., Смагин А.А. Система автоматизированного проектирования и моделирования 3D ГИС // Вестник НГИЭИ. – 2017. – № 6 (73). – С. 18–31.