УДК 519.683

В.Г. Акуловский

НЕКОТОРЫЕ АСПЕКТЫ ФОРМАЛИЗАЦИИ АРХИТЕКТУРНОГО ЭТАПА РАЗРАБОТКИ АЛГОРИТМОВ

Предложен подход к формализации некоторых аспектов архитектурного этапа разработки алгоритмов, в рамках которого решаются задачи представления алгоритма в виде совокупности подсистем, спецификации данных, обрабатываемых подсистемами, и спецификации информационных связей между подсистемами.

Введение

Этап архитектурного проектирования в процесс разработки алгоритмов играет ключевую роль так как от качества его реализации в существенной мере зависят все последующие этапы и, в конечном счете, качество программного продукта. На этом этапе решаются две основные взаимосвязанные задачи [1]:

1) определяется состав подсистем, образующих алгоритм, и специфицируются обрабатываемые и продуцируемые этими подсистемами данные;

2) специфицируются взаимосвязи между этими подсистемами.

При этом уровень формализации данного этапа проектирования явно недостаточен. В большинстве случаев проектирование ведется с использованием графических средств (граф - схемы, диаграммы и т.п.), при этом информационные связи внутри алгоритма оформляются в виде отдельного документа. Наличие нескольких документов, отражающих различные аспекты проекта, может служить источником ошибок, связанных с несоответствиями в этих документах.

Цель данной работы – решение двух названных задач в рамках единого формального аппарата. Кроме этого, наряду с требованием к процессу проектирования алгоритмов, заключающемся в необходимости обеспечения соответствия известным принципам: формальности, абстрактности, иерархической упорядоченности и т.д., приведем ограничение, контроль удовлетворения которого входит в состав решаемых задач.

Ограничение. Все специфицированные в алгоритме данные должны быть использованы, а все используемые – специфицированы.

В качестве формального аппарата, для проектирования алгоритмов на упомянутом этапе, выберем расширенную алгебру алгоритмов (САА-Р) [2], которая является расширением известной алгебры алгоритмов Глушкова [3, 4]. А в качестве метода разработки – метод структурного проектирования программ (МСПП) [5, 6], основанный на декомпозиции алгоритмов и базирующийся на выбранном формальном аппарате. Отметим, что использование МСПП, позволяет следовать вышеприведенным принципам.

Решение задачи формализации взаимодействия операторов и данных в рамках САА-Р, выполненное в [7], послужило предпосылкой к данной работе.

Таким образом, будем решать задачи, характерные для выбранного этапа разработки алгоритмов, повышая уровень его формализации за счет использования алгебраического аппарата и обеспечивая удовлетворение вышеприведенного ограничения.

Исходные определения и используемая терминология

Многие из приведенных далее определений были ранее даны в работах [7, 8], однако в данном случае они будут формулироваться в соответствии со спецификой решаемой задачи.

Используемый формальный аппарат представляет собой систему алгоритмических алгебр – $\langle U, L, \Omega \rangle$, где U – множество операторов, L – множество логических условий, Ω – сигнатура операций, со-

стоящая из логических операций – Ω_1, принимающих значения на множестве L, и операций – Ω_2, принимающих значения на множестве операторов U.

Операторы определим следующим образом.

Определение 1. Операторы, с помощью которых строится алгоритм:

$$(D)A(D') \in U,$$

где D – входные данные (область определения), D' – выходные данные (область значений). Эти операторы, обрабатывая входные данные D, изменяют их (или некоторое их подмножество) и порождают новые данные, формируя таким образом множество выходных данных D'.

Отметим, что запись оператора в виде $(D)A(D')$ позволяет специфицировать обрабатываемые этим оператором данные.

Из множества операций САА-Р в данном случае воспользуемся единственной операцией композиции, которую (с учетом определения 1) определим так.

Определение 2. Операция композиции (обозначается символом "*") двух операторов $(D_A)A(D'_A)*(D_B)B(D'_B)$ означает, что оператор $(D_A)A(D'_A)$ предшествует оператору $(D_B)B(D'_B)$, а оператор $(D_B)B(D'_B)$ следует за оператором $(D_A)A(D'_A)$, т. е. последовательно выполняются сначала оператор $(D_A)A(D'_A)$, а затем оператор $(D_B)B(D'_B)$.

Основной подход к процессу проектирования в рамках МСПП состоит в декомпозиции операторов, т. е. в представлении их в виде регулярных схем (РС), которые с учетом определений 1 и 2 введем следующим образом.

Определение 3. Регулярной схемой (РС) будем называть выражение вида

$$(D)A(D') = (D_1)A_1(D'_1)*(D_2)A_2(D'_2)*\ldots$$
$$\ldots*(D_i)A_i(D'_i)*\ldots*(D_n)A_n(D'_n)$$

(где нижний индекс – порядковый номер оператора), в котором операторы $A_1, A_2, \ldots, A_i, \ldots, A_n$ выполняются последовательно и их суммарная функциональность адекватна функциональности оператора A. Оператор $(D)A(D')$ будем называть исходным, а любой оператор $(D_i)A_i(D'_i)$ – результирующим. Специфицированные для исходных и результирующих операторов данные удовлетворяют следующим соотношениям:

$$D = \bigcup_{j=1}^{n} D_j \text{ и } D' = \bigcup_{j=1}^{n} D'_j.$$

Как уже отмечалось, одной из задач, решаемых на рассматриваемом этапе разработки, является спецификация информационных связей. Учитывая особенности этой задачи, абстрагируясь от того, что данные заносятся в память и читаются из неё, будем в процессе изложения говорить, что данные поступают на вход (появляются на входе) оператора, появляются на его выходе и передаются с выхода одного оператора на вход другого. Будем полагать, что данные передаются от некоторого оператора к одному или нескольким следующим за данным оператором, т. е. передаются в соответствии с определениями 2 и 3 слева направо.

Чтобы осуществить требуемую спецификацию связей, определим понятие связанных операторов.

Определение 4. Операторы $(D_i)A_i(D'_i)$ и $(D_j)A_j(D'_j)$ (где i<j), входящие в РС

$$(D)A(D') = (D_1)A_1(D'_1)*\ldots*(D_j)A_j(D'_j)*\ldots*(D_k)A_k(D'_k),$$

связаны, если для них выполняется соотношение: $D'_i \cap D_j \neq \varnothing$, т. е. некоторое подмножество выходных данных оператора $(D_i)A_i(D'_i)$ поступает на вход оператора $(D_j)A_j(D'_j)$. Будем говорить, что множество данных ${}_i\vec{D}_j = D'_i \cap D_j$ связывает операторы A_i и A_j (на что указывают используемые индексы) и эти данные ${}_i\vec{D}_j$ будем называть связывающими.

Из определения 4 следует, что связывающие данные образуют информационные связи в РС.

Разработка алгоритма

Будем исходить из того, что на этапах, предшествовавших архитектурному этапу проектирования алгоритма, разработаны спецификации решаемой задачи. В результате чего мы располагаем спецификациями исходных ${}^{исх}D$, результирующих ${}^{рез}D$ данных и перечнем функций, преобразующих первые во вторые.

Названные данные определим следующим образом.

Определение 5. Исходные – это определенные (инициализированные), т. е. имеющие некоторые начальные значения данных, или данные, вводимые с внешних устройств (клавиатуры, магнитных дисков, аналого-цифровых преобразователей и т.д.).

Определение 6. Результирующие – это данные, являющиеся результатом работы программной системы, отображаемые (например, на экране дисплея) и/или сохраняемые на внешних устройствах.

Очевидно, что практически в любом алгоритме, помимо вышеопределенных, присутствуют и другие данные, которые назовем промежуточными и определим следующим образом.

Определение 7. Промежуточные – это данные, отсутствующие в спецификации задачи, но необходимые для преобразования исходных данных в результирующие, так как они являются носителями промежуточных результатов, получаемых на пути от исходных к результирующим данным.

Поскольку среди используемых алгоритмом данных различают глобальные и локальные, определим их следующим образом.

Определение 8. Локальными будем называть данные, доступные только внутри результирующих операторов, входящих в РС, вне которых они не доступны и не специфицированы.

Определение 9. Глобальными будем называть данные, специфицированные для исходного оператора и доступные всем результирующим операторам, входящим в РС. При декомпозиции результирующих операторов, локализованные в них данные играют роль глобальных на следующем уровне декомпозиции. Глобальные (и только глобальные) данные на выходах операторов являются (в соответствии с определениями 3, 4, 8) связывающими.

После определения обрабатываемых алгоритмом данных уместно определить понятие алгоритм.

Определение 10. Алгоритмом назовем первый и единственный оператор на нулевом уровне декомпозиции, записанный в виде $(D_1^0, {}^{исх}D_1^0,)\mathrm{A}_1^0({}^{рез}D_1^0)$ (верхний индекс номер уровня декомпозиции, а нижний порядковый номер оператора в РС), преобразующий посредством промежуточных данных D_1^0 исходные данные ${}^{исх}D_1^0$ в результирующие ${}^{рез}D_1^0$. Причем множество глобальных (см. определение 9) данных D_1^0 – это спецификация всех (на входе и выходе) промежуточных данных, обрабатываемых результирующими операторами.

Предложенную в определении 10 смысловую нагрузку индексы будут нести с некоторыми дополнениями в процессе всего дальнейшего изложения.

Используя определения 3, 5 – 7, 10, выполним первый шаг декомпозиции алгоритма, с учетом того, что алгоритм не содержит на входе и выходе связующих данных, так как является единственным оператором на нулевом уровне декомпозиции. При этом все образующие его операторы, в соответствии с определением 4, такие данные имеют (могут иметь при наличии информационных связей между ними).

Запишем РС в виде:

$$
\begin{aligned}
&(D_1^0, {}^{исх}D_1^0)\mathrm{A}_1^0({}^{рез}D_1^0) = \\
&\quad = (D_1^1, {}^{исх}D_1^1)A_1^1({}^{рез}D_1^1, {}_1D'^1)* \\
&\quad *(D_2^1, {}^{исх}D_2^1)A_2^1({}^{рез}D_2^1, {}_2D'^1)*... \\
&\quad ...*(D_i^1, {}^{исх}D_i^1)A_i^1({}^{рез}D_i^1, {}_iD'^1)*... \\
&\quad ...*(D_n^1, {}^{исх}D_n^1)A_n^1({}^{рез}D_n^1).
\end{aligned}
\qquad (1)
$$

Отметим, что особенности в расположении индексов для всех промежуточных данных ${}_j D'^1$ (пример для j-го результирующего оператора) будут использованы и понятны из дальнейшего изложения.

Рассмотрим свойства, которыми обладает построенная на первом шаге декомпозиции РС и используемые в ней данные. Причем, результирующие операторы в контексте данной работы в дальнейшем изложении будем называть подсистемами.

Из определений 3, 9 и 10 следует, что РС (1) представляет собой совокупность подсистем, образующих алгоритм, причем эта совокупность реализует все функции по преобразованию всех исходных данных во все результирующие, а данные, специфицированные для алгоритма являются глобальными для всех образующих его подсистем. При этом, первая подсистема не содержит на входе, а последняя (n-ая) – на выходе связывающих данных, так как первой не предшествует, а за последней не следует ни одной подсистемы.

Для исходных, результирующих и промежуточных данных (в соответствии с определением 3) имеют место следующие свойства:

$$ {}^{исх}D_1^0 = \bigcup_{i=1}^{n} {}^{исх}D_i^1 \ , \qquad (2) $$

$$ {}^{рез}D_1^0 = \bigcup_{i=1}^{n} {}^{рез}D_i^1 \ , \qquad (3) $$

$$ D_1^0 = \bigcup_{i=1}^{n} D_i^1 \cup \bigcup_{i=1}^{n-1} {}_i D'^1 \ , \qquad (4) $$

т. е. все исходные, результирующие и промежуточные данные распределяются между подсистемами, образующими алгоритм.

Наличие у алгоритма на первом шаге декомпозиции свойств 2 – 4 позволяют удовлетворить ограничение, сформулированное во введении.

Реализовав первый шаг декомпозиции и рассмотрев свойства данных, характерных для этого шага, перейдем к решению второй задачи – спецификации связей между подсистемами.

В общем случае каждая подсистема, входящая в РС, может быть связана со всеми следующими за ней и всеми предшествующими ей подсистемами. То есть, у любой подсистемы множество выходных данных содержит данные, связывающие её со всеми следующими за ней подсистемами, а множество входных данных включает в себя данные, связывающие её со всеми предшествующими ей подсистемами.

Такой общий случай и будем рассматривать, предварительно дополнив систему обозначений (с учетом обозначений, использованных в определении 4) следующим образом. Связывающие данные на входе подсистемы будем обозначать ${}_k\vec{D}_j^1$, а на выходе – ${}_l\vec{D}_p^1$, причем левый нижний индекс – порядковый номер подсистемы в РС будем называть адресом источника, а правый нижний индекс – адресом приемника данных. Заметим, что для реализации такой системы обозначений в (1) было использовано соответствующее расположение индексов.

Перепишем РС (1) с учётом введенных обозначений:

$$
\begin{aligned}
&(D_1^0, {}^{исх}D_1^0)A_1^0({}^{рез}D_1^0) = \\
&= (D_1^1, {}^{исх}D_1^1)A_1^1({}^{рез}D_{1,1}^1\vec{D}_{2,1}^1\vec{D}_3, \ldots, {}_1\vec{D}_j^1, \ldots, {}_1\vec{D}_n^1) * \\
&*(D_2^1, {}^{исх}D_{2,1}^1\vec{D}_2^1,)A_2^1({}^{рез}D_{2,2}^1\vec{D}_3^1, \ldots, {}_2\vec{D}_j^1, \ldots, {}_2\vec{D}_n^1) * \ldots \qquad (5) \\
&.*(D_i^1, {}^{исх}D_{i,1}^1\vec{D}_j^1, \ldots, {}_{j-1}\vec{D}_j^1)A_j^1({}^{рез}D_{j,j}^1\vec{D}_{j+1}^1, \ldots, {}_j\vec{D}_n^1) * \\
&\ldots *(D_n^1, {}^{исх}D_{n,1}^1\vec{D}_{n,2}^1\vec{D}_n, \ldots, {}_j\vec{D}_n^1, \ldots, {}_{n-1}\vec{D}_n^1)A_n^1({}^{рез}D_n^1)
\end{aligned}
$$

?

В построенной РС не только специфицированы данные, но и все “источники” и “приемники” данных поставлены в однозначное соответствие, т. е. любому ${}_l\vec{D}_p^1$ соответствует ${}_l\overleftarrow{D}_p^1$, и, таким образом, выполняется

$$\bigcup_{l=1}^{n-1}\bigcup_{k=2}^{n} {}_l\vec{D}_k^i = \bigcup_{l=1}^{n-1}\bigcup_{k=2}^{n} {}_l\overleftarrow{D}_k^i . \qquad (6)$$

Данная РС сохраняет свойства (2) и (3) в неизменном виде, а свойство (4) трансформируется к виду:

$$D_1^0 = \bigcup_{p=1}^{n} D_p^1 \cup \bigcup_{l=1}^{n-1}\bigcup_{k=2}^{n} {}_l\vec{D}_k^i . \qquad (7)$$

В связи с наличием этих свойств, ограничение удовлетворяются и в данном случае.

Переходя ко второму шагу проектирования, рассмотрим особенности использования локальных данных. На первом шаге разработки мы не включали в спецификацию локальные данные по двум причинам. Во-первых, в соответствии с определением 8 они доступны только внутри результирующих операторов и, учитывая это, не влияют на процесс разработки на первом уровне декомпозиции. Во-вторых, их включение в спецификацию вступило бы в противоречие с ограничением.

Рассмотрение второго шага начнем с декомпозиции і-й подсистемы из (1), которую запишем в виде РС, где на входе исходной подсистемы специфицируем локальные данные:

$$\begin{aligned}(D_i^1, {}^{исх}D_i^1, {}^{л}D_i^1)A_i^1({}^{рез}D_i^1, {}_iD'^1) &= \\ = (D_1^2, {}^{исх}D_1^2)A_1^2({}^{рез}D_1^2, {}_1D'^2)* &\\ *(D_2^2, {}^{исх}D_2^2)A_2^2({}^{рез}D_2^2, {}_2D'^2)*... &\\ ...*(D_j^2, {}^{исх}D_j^2)A_j^2({}^{рез}D_j^2, {}_jD'^2)*... &\\ ...*(D_k^2, {}^{исх}D_k^2)A_k^2({}^{рез}D_k^2, {}_kD'^2). &\end{aligned} \qquad (8)$$

Отметим, что локальные на первом уровне декомпозиции данные ${}^{л}D_i^1$ в соответствии с определением 9 для второго уровня уже играют роль глобальных, то есть для любого D_j^2 (j пробегает значения от 1 до k) выполняется $D_j^2 \subseteq D_i^1 \cup {}^{л}D_i^1$ и множество данных ${}^{л}D_i^1$ в качестве глобального распределяется между всеми подсистемами, входящими в РС.

Полученная после спецификации локальных данных РС сохраняет свойства (2) и (3), которые для данного уровня записываются в виде:

$${}^{исх}D_i^1 = \bigcup_{j=1}^{k} {}^{исх}D_j^2 , \qquad (9)$$

$${}^{рез}D_i^1 = \bigcup_{j=1}^{k} {}^{рез}D_j^2 . \qquad (10)$$

Кроме этих свойств РС (8), в связи с использованием локальных данных, обладает следующими свойствами:

$$D_i^1 \cup {}^{л}D_i^1 = \bigcup_{j=1}^{k} D_j^2 , \qquad (11)$$

$${}_iD'^1 = \bigcup_{j=1}^{k} {}_jD'^2 , \qquad (12)$$

из которых следует свойство аналогичное (4)

$$D_i^1 \cup {}^{л}D_i^1 \cup {}_jD'^1 = \bigcup_{j=1}^{k}(D_j^2 \cup {}_jD'^2), \qquad (13)$$

и, таким образом, ограничение выполняется и в данном случае.

Применив такой подход ко всем подсистемам первого уровня, получим совокупность РС, которую будем называть слоем алгоритма. При рассмотрении совокупности результирующие подсистемы, образующие первый слой алгоритма, будем использовать сквозную нумерацию операторов внутри слоя, и в этом случае нижний индекс будет определять порядковый номер оператора в рассматриваемом слое.

Учитывая вышеизложенное, первый слой алгоритма запишем в виде:

$$(D_1^1, {}^{исх}D_1^1, {}^{л}D_1^1)A_1^1({}^{рез}D_1^1, {}_1D'^1) = (D_1^2, {}^{исх}D_1^2)A_1^2({}^{рез}D_1^2, {}_1D'^2) *$$
$$*(D_2^2, {}^{исх}D_2^2)_1A_2^2({}^{рез}D_2^2, {}_2D'^2) * ...$$
$$...*(D_j^2, {}^{исх}D_j^2)A_j^2({}^{рез}D_j^2, {}_jD'^2) * ...$$
$$...*(D_{k_1}^2, {}^{исх}D_{k_1}^2)A_{k_1}^2({}^{рез}D_{k_1}^2, {}_{k_1}D'^2)$$

$$(D_2^1, {}^{исх}D_2^1, {}^{л}D_2^1)A_2^1({}^{рез}D_2^1, D_2'^1) = (D_{k_1+1}^2, {}^{исх}D_{k_1+1}^2)A_{k_1+1}^2({}^{рез}D_{k_1+1}^2, {}_{k_1+1}D'^2) *$$
$$*(D_{k_1+2}^2, {}^{исх}D_{k_1+2}^2)A_{k_1+2}^2({}^{рез}D_{k_1+2}^2, {}_{k_1+2}D'^2) * ...$$
$$...*(D_{k_1+j}^2, {}^{исх}D_{k_1+j}^2)A_{k_1+j}^2({}^{рез}D_{k_1+j}^2, {}_{k_1+j}D'^2) * ...$$
$$...*(D_{k_2}^2, {}^{исх}D_{k_2}^2)A_{k_2}^2({}^{рез}D_{k_2}^2, {}_{k_2}D'^2) \qquad (14)$$

……

$$(D_i^1, {}^{исх}D_i^1, {}^{л}D_i^1)A_i^1({}^{рез}D_i^1, {}_iD'^1) = (D_{k_{i-1}+1}^2, {}^{исх}D_{k_{i-1}+1}^2)A_{k_{i-1}+1}^2({}^{рез}D_{k_{i-1}+1}^2, {}_{k_1+1}D'^2) *$$
$$*(D_{k_{i-1}+2}^2, {}^{исх}D_{k_{i-1}+2}^2)A_{k_{i-1}+2}^2({}^{рез}D_{k_{i-1}+2}^2, {}_{k_{i-1}+2}D'^2) * ...$$
$$...*(D_{k_{i-1}+j}^2, {}^{исх}D_{k_{i-1}+j}^2)A_{k_{i-1}+j}^2({}^{рез}D_{k_{i-1}+j}^2, {}_{k_{i-1}+j}D'^2) * ...$$
$$...*(D_{k_i}^2, {}^{исх}D_{k_i}^2)A_{k_i}^2({}^{рез}D_{k_i}^2, {}_{k_i}D'^2)$$

……

$$(D_n^1, {}^{исх}D_n^1, {}^{л}D_n^1)A_n^1({}^{рез}D_n^1, {}_nD'^1) = (D_{k_{n-1}+1}^2, {}^{исх}D_{k_{n-1}+1}^2)A_{k_{n-1}+1}^2({}^{рез}D_{k_{n-1}+1}^2, {}_{k_{n-1}+1}D'^2) *$$
$$*(D_{k_{n-1}+2}^2, {}^{исх}D_{k_{n-1}+2}^2)A_{k_{n-1}+2}^2({}^{рез}D_{k_{n-1}+2}^2, {}_{k_{n-1}+2}D'^2) * ...$$
$$...*(D_{k_{n-1}+j}^2, {}^{исх}D_{k_{n-1}+j}^2)A_{k_{n-1}+j}^2({}^{рез}D_{k_{n-1}+j}^2, {}_{k_{n-1}+j}D'^2) * ...$$
$$...*(D_{k_n}^2, {}^{исх}D_{k_n}^2)A_{k_n}^2({}^{рез}D_{k_n}^2),$$

где n – количество исходных операторов и, соответственно, количество РС;

k_n – общее количество подсистем, образующих слой.

Обобщим свойства (9) – (13) на случай слоя алгоритма:

$$\bigcup_{i=1}^{n} {}^{исх}D_i^1 = \bigcup_{j=1}^{k_n} {}^{исх}D_j^2 \ , \qquad (15)$$

$$\bigcup_{i=1}^{n} {}^{рез}D_i^1 = \bigcup_{j=1}^{k_n} {}^{рез}D_j^2 \ , \qquad (16)$$

$$\bigcup_{i=1}^{n} (D_i^1 \cup {}^{л}D_i^1) = \bigcup_{j=1}^{k_n} D_j^2 \ , \qquad (17)$$

$$\bigcup_{i=1}^{n} {}_iD'^1 = \bigcup_{j=1}^{k_n} {}_jD'^2 \qquad (18)$$

$$\bigcup_{i=1}^{n} (D_i^1 \cup {}^{л}D_i^1 \cup {}_iD'^1) = \bigcup_{j=1}^{k} (D_j^2 \cup {}_jD'^2) \qquad (19)$$

и, исходя из полученных свойств, сделаем вывод о том, что ограничение выполняется и для слоя алгоритма.

Теперь перейдем к спецификации информационных связей в построенном слое алгоритма, для чего перепишем выражение (14) в виде, аналогичном (5).

$$(D_1^1, {}^{исх}D_1^1, {}^{л}D_1^1)A_1^1({}^{рез}D_1^1, {}_1D'^1) =$$
$$= (D_1^2, {}^{исх}D_1^2)A_1^2({}^{рез}D_1^2, {}_1\vec{D}_2^2, {}_1\vec{D}_3^2, ..., {}_1\vec{D}_j^2, ..., {}_1\vec{D}_{k_1}^2, ..., {}_1\vec{D}_{k_i}^2, ..., {}_1\vec{D}_{k_n-1}^2, {}_1\vec{D}_{k_n}^2) *$$
$$*(D_2^i, {}_1\tilde{D}_2^2, {}^{исх}D_2^2)A_2^2({}^{рез}D_2^2, {}_2\vec{D}_3^2, {}_2\vec{D}_4^2, ..., {}_2\vec{D}_j^2, ..., {}_2\vec{D}_{k_1}^2, ..., {}_2\vec{D}_{k_i}^2, ..., {}_2\vec{D}_{k_n-1}^2, {}_2\vec{D}_{k_n}^2) * ...$$
$$...*(D_j^i, {}_1\tilde{D}_j^i, ..., {}_2\tilde{D}_j^i, ..., {}_{j-1}\tilde{D}_j^i, {}^{исх}D_j^i)A_j^i({}^{рез}D_j^2, {}_j\vec{D}_{j+1}^2, ..., {}_j\vec{D}_{k_1}^2, ..., {}_j\vec{D}_{k_i}^2, ..., {}_j\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_1}^2, {}_1\tilde{D}_{k_1}^2, ..., {}_2\tilde{D}_{k_1}^2, ..., {}_{k_1-1}\tilde{D}_{k_1}^2, {}^{исх}D_{k_1}^2)A_{k_1}^2({}^{рез}D_{k_1}^2, {}_{k_1}\vec{D}_{k_1+1}^2, ..., {}_{k_1}\vec{D}_{k_i}^2, ..., {}_{k_1}\vec{D}_{k_n}^2)$$

$$(D_2^1, {}^{исх}D_2^1, {}^{л}D_2^1)A_2^1({}^{рез}D_2^1, {}_2D'^1) =$$
$$= (D_{k_1+1}^2, {}_1\tilde{D}_{k_1+1}^2, {}_2\tilde{D}_{k_1+1}^2, ..., {}_{k_1}\tilde{D}_{k_1+1}^2, {}^{исх}D_{k_1+1}^2)A_{k_1+1}^2({}^{рез}D_{k_1+1}^2, {}_{k_1+1}\vec{D}_{k_1+2}^2, ..., {}_{k_1+1}\vec{D}_{k_i}^2, ..., {}_{k_1+1}\vec{D}_{k_n}^2) *$$
$$*(D_{k_1+2}^2, {}_1\tilde{D}_{k_1+2}^2, {}_2\tilde{D}_{k_1+2}^2, ..., {}_{k_1+1}\tilde{D}_{k_1+2}^2, {}^{исх}D_{k_1+2}^2)A_{k_1+2}^2({}^{рез}D_{k_1+2}^2, {}_{k_1+2}\vec{D}_{k_1+3}^2, ..., {}_{k_1+2}\vec{D}_{k_i}^2, ..., {}_{k_1+2}\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_1+j}^2, {}_1\tilde{D}_{k_1+j}^2, {}_2\tilde{D}_{k_1+j}^2, ..., {}_{k_1+j-1}\tilde{D}_{k_1+j}^2, {}^{исх}D_{k_1+j}^2)A_{k_1+j}^2({}^{рез}D_{k_1+j}^2, {}_{k_1+j}\vec{D}_{k_1+j+1}^2, ..., {}_{k_1+j}\vec{D}_{k_i}^2, ..., {}_{k_1+j}\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_2}^2, {}_1\tilde{D}_{k_2}^2, {}_2\tilde{D}_{k_2}^2, ..., {}_{k_2-1}\tilde{D}_{k_2}^2, {}^{исх}D_{k_2}^2)A_{k_2}^2({}^{рез}D_{k_2}^2, {}_{k_2}\vec{D}_{k_2+1}^2, {}_{k_2}\vec{D}_{k_2+2}^2, ..., {}_{k_2}\vec{D}_{k_2+j}^2, ..., {}_{k_2}\vec{D}_{k_i}^2, ..., {}_{k_2}\vec{D}_{k_n}^2)$$

……

$$(D_i^1, {}^{исх}D_i^1, {}^{л}D_i^1)A_i^1({}^{рез}D_i^1, {}_iD'^1) =$$
$$= (D_{k_{i-1}+1}^2, {}_1\tilde{D}_{k_{i-1}+1}^2, {}_2\tilde{D}_{k_{i-1}+1}^2, ..., {}_{k_{i-1}}\tilde{D}_{k_{i-1}+1}^2, {}^{исх}D_{k_{i-1}+1}^2)A_{k_{i-1}+1}^2({}^{рез}D_{k_{i-1}+1}^2, {}_{k_{i-1}+1}\vec{D}_{k_{i-1}+2}^2, ..., {}_{k_{i-1}+1}\vec{D}_{k_i}^2, ..., {}_{k_{i-1}+1}\vec{D}_{k_n}^2) *$$
$$*(D_{k_{i-1}+2}^2, {}_1\tilde{D}_{k_{i-1}+2}^2, {}_2\tilde{D}_{k_{i-1}+2}^2, ..., {}_{k_{i-1}+1}\tilde{D}_{k_{i-1}+2}^2, {}^{исх}D_{k_{i-1}+2}^2)A_{k_{i-1}+2}^2({}^{рез}D_{k_{i-1}+2}^2, {}_{k_{i-1}+2}\vec{D}_{k_{i-1}+3}^2, ..., {}_{k_{i-1}+2}\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_{i-1}+j}^2, {}_1\tilde{D}_{k_{i-1}+j}^2, {}_2\tilde{D}_{k_{i-1}+j}^2, ..., {}_{k_{i-1}+j-1}\tilde{D}_{k_{i-1}+j}^2, {}^{исх}D_{k_{i-1}+j}^2)A_{k_{i-1}+j}^2({}^{рез}D_{k_{i-1}+j}^2, {}_{k_{i-1}+j}\vec{D}_{k_{i-1}+j+1}^2, ..., {}_{k_{i-1}+1}\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_i}^i, {}_1\tilde{D}_{k_i}^2, {}_2\tilde{D}_{k_i}^2, ..., {}_{k_i-1}\tilde{D}_{k_i}^2, {}^{исх}D_{k_i}^i)A_{k_i}^i({}^{рез}D_{k_i}^i, {}_{k_i}\vec{D}_{k_i+1}^2, ..., {}_{k_i}\vec{D}_{k_i+2}^2, ..., {}_{k_i}\vec{D}_{k_i+j}^2, ..., {}_{k_i}\vec{D}_{k_{n-1}}^2, ..., {}_{k_i}\vec{D}_{k_n}^2)$$

……

$$(D_n^{i-1}, {}^{исх}D_n^{i-1}, {}^{л}D_n^{i-1})A_n^{i-1}({}^{рез}D_n^{i-1}, {}_nD'^{i-1}) =$$
$$= (D_{k_{n-1}+1}^2, {}_1\tilde{D}_{k_{n-1}+1}^2, {}_2\tilde{D}_{k_{n-1}+1}^2, ..., {}_{k_{n-1}}\tilde{D}_{k_{n-1}+1}^2, {}^{исх}D_{k_{n-1}+1}^2)A_{k_{n-1}+1}^2({}^{рез}D_{k_{n-1}+1}^2, {}_{k_{n-1}+1}\vec{D}_{k_{n-1}+2}^2, ..., {}_{k_{n-1}+1}\vec{D}_{k_n}^2) *$$
$$*(D_{k_{n-1}+2}^2, {}_1\tilde{D}_{k_{n-1}+2}^2, {}_2\tilde{D}_{k_{n-1}+2}^2, ..., {}_{k_{n-1}+1}\tilde{D}_{k_{n-1}+2}^2, {}^{исх}D_{k_{n-1}+2}^2)A_{k_{n-1}+2}^2({}^{рез}D_{k_{n-1}+2}^2, {}_{k_{n-1}+2}\vec{D}_{k_{n-1}+3}^2, ..., {}_{k_{n-1}+2}\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_{n-1}+j}^2, {}_1\tilde{D}_{k_{n-1}+j}^2, ..., {}_{k_{n-1}+j-1}\tilde{D}_{k_{n-1}+j}^2, {}^{исх}D_{k_{n-1}+j}^2)A_{k_{n-1}+j}^2({}^{рез}D_{k_{n-1}+j}^2, {}_{k_{n-1}+j}\vec{D}_{k_{n-1}+j+1}^2, ..., {}_{k_{n-1}+j}\vec{D}_{k_n}^2) * ...$$
$$...*(D_{k_n}^2, {}_1\tilde{D}_{k_n}^2, {}_2\tilde{D}_{k_n}^2, ..., {}_{k_n-1}\tilde{D}_{k_n}^2, {}^{исх}D_{k_n}^2)A_{k_n}^2({}^{рез}D_{k_n}^2)$$

Для данного случая свойства (15),(16) сохраняются неизменными, а остальные свойства обобщим на случай слоя со специфицированными информационными связями.

Свойство (6) приобретает вид

$$\bigcup_{l=1}^{k_{n-1}}\bigcup_{p=2}^{k_n} {}_l\vec{D}_p^i = \bigcup_{l=1}^{k_{n-1}}\bigcup_{p=2}^{k_n} {}_l\tilde{D}_p^i,$$

а свойства (17 – (19) трансформируются к виду:

$$\bigcup_{i=1}^{n}(D_i^1 \cup {}^{л}D_i^1) = \bigcup_{j=1}^{k_n} D_j^2 \cup \bigcup_{l=1}^{k_{n-1}}\bigcup_{p=2}^{k_n} {}_l\tilde{D}_p^i,$$

$$\bigcup_{i=1}^{n} {}_iD'^1 = \bigcup_{l=1}^{k_{n-1}}\bigcup_{p=2}^{k_n} {}_l\vec{D}_p^i,$$

$$\bigcup_{i=1}^{n}(D_i^1 \cup {}^{л}D_i^1 \cup {}_iD'^1) = \bigcup_{i=1}^{n}(D_j^2 \cup \bigcup_{l=1}^{k_{n-1}}\bigcup_{p=2}^{k_n} {}_l\vec{D}_p^i).$$

Для локальных данных в первом слое алгоритма в рамках всех РС выполняются соотношения, которые запишем для случая i-й РС:

$${}^{л}D_i^1 \subseteq \bigcup_{l=k_{i-1}+1}^{k_i-1}\bigcup_{p=k_{i-1}+2}^{k_i} {}_l\vec{D}_p^i$$

или в соответствии с (6)

$$ {}^{л}D_i^1 \subseteq \bigcup_{l=k_{i-1}+1}^{k_i-1} \bigcup_{p=k_{i-1}+2}^{k_i} {}_l\vec{D}_p^i , $$

т. е. в соответствии с определениями 8 и 9 локальные данные играют роль глобальных только в рамках одной (i-й в данном случае) РС.

Полученные свойства позволяют утверждать, что ограничение выполняется для каждой РС, входящей в данный слой, и для всего слоя алгоритма.

Заметим, что в последнем случае, как и во всех предыдущих, первая подсистема, для которой нет предшествующих, не содержит на входе, а последняя подсистема, для которой нет последующих подсистем, не содержит на выходе связывающих данных.

Таким образом, нам удалось специфицировать связи между подсистемами в рамках первого слоя алгоритма и сформулировать свойства данных в этом слое.

Очевидно, что процесс декомпозиции повторяется на каждом шаге разработки, и структура второго и последующих слоев будет полностью повторять структуру первого слоя, сохраняя при этом все вышесформулированные свойства. Различие между слоями алгоритма будет состоять только в том, что каждый последующий слой будет “толще” предыдущего за счет роста числа РС.

Процесс декомпозиции на данном этапе продолжается до того момента, когда будут выделены все подсистемы, образующие проектируемую программную систему. В результате такого построения получим некоторое количество слоёв алгоритма, где каждый, ниже расположенный, будет носить все менее абстрактный (более детализованный) характер и, как вышеотмечено, будет достигнуто соответствие принципам, упомянутым во введении. Еще раз так же отметим, что сформулированное во введении ограничение в рамках данного подхода удовлетворяется при выполнении всех шагов архитектурного этапа разработки алгоритмов.

При рассмотрении данного этапа мы брали самый общий случай, когда каждая подсистема, входящая в РС или слой алгоритма, связана со всеми предшествующими и всеми следующими за ней. На практике такие случаи встречаются достаточно редко, т. е. обычно не все подсистемы связаны друг с другом и имеют место не связанные (независимые) с другими подсистемами. В связи с чем заметим, что любое из множеств ${}^{исх}D_i^m$, ${}^{рез}D_i^m$, D_i^m, ${}_iD'^m$ (в любом сочетании, но не все сразу) может быть пустым и, таким образом, в конкретных разработках допустимы многочисленные частные случаи подсистем (например: $(D_i^1, {}^{исх}D_i^1, {}^{л}D_i^1)A_i^1({}_iD'^1)$, $(D_i^1, {}^{л}D_i^1)A_i^1({}^{рез}D_i^1, {}_iD'^1)$, $(D_i^1, {}^{л}D_i^1)A_i^1({}_iD'^1)$ и т.д.), обладающих приведенными в работе свойствами.

Заключение

Таким образом, предложен подход к архитектурному этапу разработки алгоритмов, по завершению которого разработчик выявляет структурированное представление о программной системе, т. е. определяет состав и функции подсистем, а так же взаимосвязи между всеми подсистемами.

Традиционно процесс декомпозиции реализуется двумя путями. При первом (от управления) – декомпозируются операторы и в соответствии с понижением уровня их функциональности детализуются обрабатываемые оператором данные. При втором (от данных) – детализуются данные и, в соответствии с полученной степенью детализации, определяется функциональность операторов для их обработки.

Подход, предложенный в данной работе, основывается на согласованной декомпозиции операторов и детализации данных, что позволяет рассчитывать на повышение качества разрабатываемого алгоритма. Расчет строится на том, что на каждом шаге разработки выбирается наиболее эффективный в данном случае способ, так как при этом может быть проанализирован и учтен возможный результат использования альтернативного варианта.

Анализ информационных связей в алгоритмах является перспективным

средством контроля корректности и преобразования алгоритмов. В частности, приведенные в работе свойства данных можно рассматривать как средство контроля корректности алгоритма с точки зрения обработки данных на всех уровнях декомпозиции алгоритма. Применение предложенного подхода на других этапах разработки алгоритмов и реализация указанных перспектив являются направлением дальнейших исследований.

1. *Соммервилл И.* Инженерия программного обеспечения. – М.: Издательский дом “Вильямс”, 2002. – 642 с.
2. *Акуловский В.Г.* Расширенная алгебра алгоритмов // Проблеми програмування. – 2007. – № 3. – С. 3 – 15.
3. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – К.: Наук. думка, 1978. – 319 с.
4. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – К.: Академперіодика, 2007. – 634 с.
5. *Цейтлин Г.Е., Бакулин А.В.* Многоуровневые структурированные проекты программ и их обоснование // Кибернетика и системный анализ. – 1991. – № 5. – С. 98 – 107.
6. *Ющенко Е.Л., Цейтлин Г.Е.,Грицай В.П., Терзян Т.К.* Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.
7. *Акуловский В.Г.* Формализация взаимосвязей операторов и данных в рамках расширенной алгебры алгоритмов // Кибернетика и системный анализ. – 2008. – № 6. – С. 32 – 37.
8. *Акуловский В.Г.* Некоторые подходы к контролю и преобразованию алгоритмов на основе анализа специфицируемых данных // Проблеми програмування. – 2008. – № 4. – С . 84 – 93.

Получено 16.03.2009

Об авторе:

Акуловский Валерий Григорьевич,
канд.техн.наук, доцент кафедры информационных систем и технологий
e-mail: valeryakulovskiy@rambler.ru

Место работы автора:

Академия таможенной службы Украины.
49000, Днепропетровск, ул. Дзержинского 2\4. Тел/факс. канцелярия – (0562) 45 5596
e-mail :academy@amsu.dp.ua