ЭМПИРИЧЕСКОЕ ПРЕДСКАЗАНИЕ И РАСПОЗНАВАНИЕ ОБРАЗОВ
(Вычислительные системы)
1980 год Выпуск 83

УДК 517.11.518.5

ОБ ОДНОМ ПОДХОДЕ К ТЕОРИИ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

Е.А.Жевлакова

Данная работа представляет собой теоретическое исследование в области программирования и демонстрирует возможность формализации таких понятий, которые на первый взгляд кажутся принципиально неформализуемыми. Общие соображения по поводу такой формализации содержатся в [I], а на их основе рассматривается организация общения между разработчиками сложной системы и заказчиком, для которого она делается. Проектирование сложной программы можно условно разбить на следующие этапы.

I. Прием заказа; на этом этапе заказчик формулирует задачу руководителю разработчиков, в ходе диалога уточняется ее смысл, и у руководителя разработчиков складывается предварительное представление о структуре задачи.

2. Решение задачи, т.е. организация коллектива разработчиков, создание технологии конструирования программы, собственно конструирование самой программы.

3. Убеждение заказчика в том, что написанная программа реализует поставленную им задачу.

4. Непосредственная работа с данной программой (ее эксплуатация).

5. Возможная модификация программы.

Здесь мы сосредоточимся на первом этапе и попытаемся выявить отдельные моменты, существенные для всех этапов. Некоторые понятия (как, например, опорный прецедент, диагонализация, автопродуктивность) заимствованы из [I].

Общая постановка задачи принадлежит Н.В.Белякину и Д.И.Свириденко; в основу ее положены некоторые обсуждения на методологи-

ческом семинаре Института математики, посвященном проблемам логики, семиотики и программирования.

§I. Диалог как составная часть проектирования сложной системы

Первый этап – диалог между руководителем разработчиков архитектором A и заказчиком Z – важный момент в деле построения сложной системы, так как от результата диалога существенно зависит, какой будет программа, сможет ли она удовлетворить всем требованиям заказчика. Чтобы лучше уяснить значение диалога в более широком контексте и лучше представить себе организацию всего процесса построения системы, рассмотрим сначала наивную реализацию процесса общения A и Z, которая охватывает первые три из перечисленных этапов – ведь содержание первого этапа должно как-то ориентироваться на то, что будет в дальнейшем. Приблизительно ситуацию можно описать следующим образом.

I-й этап. Z сообщает A содержание задачи в той мере, в какой он на это способен, и A каким-то образом понимает Z; конечно, взаимопонимание между A и Z может быть и иллюзорным.

2-й этап. Диалог временно прекращается, A составляет схему решения задачи, разбивает ее на некоторое число подзадач и поручает их решение отдельным разработчикам; написанные подпрограммы стыкуются.

3-й этап. A комментирует полученную программу понятным для Z образом, в результате получается модифицированная версия задачи, и диалог между A и Z возобновляется.

Z, может согласиться с предложенной программой (точнее, с комментариями к программе), но возможно, что по каким-то причинам она не устраивает Z. Тогда он сообщает об этих причинах A, A принимает их к сведению, уточняет формулировку задачи, и цикл повторяется; причем повторяться он будет до тех пор, пока Z, наконец, не примет программу, предложенную A.

Признавая нереальность только что описанной ситуации, мы все же считаем целесообразным описать этот наивный диалог более подробно, чтобы выявить отдельные моменты, существенные для разработки методологии реального диалога между заказчиком и архитектором.

Итак, предположим, что между A и Z происходит следующий диалог. Z формулирует задачу R в терминах некоторого языка $L_R(a_1, \ldots, a_n)$, где параметры $a_1, \ldots, a_n$ имеют нечеткий смысл, т.е. есть интуитивное представление о смысле a_i, $i = 1, \ldots, n$, но

само содержание a_i мыслится как необозримое – оно не поддается никакому заранее заданному точному описанию (А не обязан понимать смысл каждого a_i). Можно считать, что язык L_R – это произвольный алгоритмический язык, обогащенный нечеткими терминами $a_1,\ldots, a_n$, т.е. синтаксис L_R – четкий, а нечеткость заключается в семантике некоторых терминов. Таким образом, нечеткий смысл задачи является четкой комбинацией смыслов терминов, отдельные из которых – нечеткие. Предположим, что у Z имеется интуитивно неисчерпаемый запас K уточнений смысла R. Так как смысл R – нечеткий, то элементы K задаются некоторой достаточно точно описанной системой прецедентов и некоторыми интуитивными соображениями. В процессе диалога между Z и A смысл R как-то уточняется (соответственно уточняется и смысл параметров a_i), в результате чего A формулирует R' – уточненную задачу Z в языке $L_{R'}(c_1,\ldots,c_l)$, где термины $c_1,\ldots,c_l$ тоже нечеткие, но понятные Z и A; если отдельные a_i не понятны A, то Z разъясняет смысл a_i в терминах списка c_j, $j = 1,\ldots, l$. Z соглашается, что R' соответствует его целям (т.е. $R' \in K$). В терминах более общих рассмотрений, изложенных в [2], имеем следующее. На этом этапе диалога первоначальная задача Z выступает в роли реальности (R), а сформулированная A эта же задача выступает уже в роли теории R'. Имеем

$$\langle R, R', M \rangle, \qquad (1)$$

где $M: R \to R'$. Здесь в качестве мостика M выступает сам диалог, в ходе которого достигается относительная коммуникабельность между A и Z.

На основании R' (где R' выступает уже как реальность) A делает проект общего решения задачи T' в языке $L_{T'}(b'_1,\ldots,b'_k)$, где $b'_1,\ldots,b'_k$ некоторые термины, предположительно тоже нечеткие, но понятные для разработчиков, которыми A руководит (и для самого A, конечно; для Z термины b'_m, $m = 1,\ldots,k$, могут быть и не понятны). Если термины a_i и c_j ориентированы на понимание содержания задачи, то термины b'_m – на то, как эту задачу решать. Имеем

$$\langle R', T', M_1 \rangle, \qquad (2)$$

где $M_1: R' \to T'$ – транслятор описания задачи в "непрограммистских" терминах c_j в описание той же задачи в "программистских" терминах b'_m; T' – функциональная схема деятельности (схема технологии конструирования будущей программы).

A разбивает T' на m подзадач, решение которых поручает разработчикам. Так как термины b'_m, нечеткие, каждый из разработ-

чиков понимает их по-своему, но в процессе работы, в результате общения друг с другом и с А происходит уточнение смыслов b'_n, в итоге вырабатываются новые понятия $b_1,\ldots,b_k$. Смысл терминов b_n, $n = 1,\ldots, k$, уже достаточно четкий на оперативном уровне: каждый из разработчиков может их запрограммировать каким-то образом, а как это сделать конкретно, чтобы было приемлемо для всей задачи, разработчики договариваются между собой и с А. Таким образом, решение всех m подзадач происходит в языке $L_T(b_1,\ldots, b_k)$, где Т - программа, которую должны построить разработчики, исходя из Т'. Исследование только что описанной ситуации - задача пост - роения модели "коллективной логики" - проблема нетривиальная и актуальная, но ее рассмотрение выходит за рамки данной статьи. Здесь мы лишь отметим, что происходит переход (мостик M_2) от функциональной схемы деятельности Т' (выступающей здесь как реальность) к технологии Т (теории)

$$\langle T', T, M_2\rangle, \qquad (3)$$

где $M_2: T' \to T$ - коллективное обсуждение, согласование смыслов нечетких понятий и т.п.

Рассмотрим тройки (I)-(3), возьмем их композицию и получим

$$\langle R, T, M_3\rangle, \qquad (4)$$

где $M_3: R \to T$ есть композиция $M \circ M_1 \circ M_2$.

Хотя сейчас мы описываем первый цикл диалога, все сказанное здесь относится и к последующим циклам. Разумно считать, что теория Т, возникающая на (n+1)-м цикле диалога, представляет собой улучшенную модификацию Т, появившейся на n-м цикле. Соответственно мостик M_3 можно отождествить с оператором воссоздания теории $T - \Pi_1$ (см. [2]). К (4) можно применить оператор адаптации Π_2 (см. [2]), в результате получится

$$\langle R^*, T, \Pi_2\rangle, \qquad (5)$$

где $\Pi_2: T \to R^*$; R^* - комментарий к программе; Т - уточненное описание задачи R, которое предъявляется Z. Мостик Π_2 - это нор - мативы функционирования программы, технология ее эксплуатации, обоснование того, что R^* является действительно уточнением R. Теперь предположим, что создание троек (I)-(3), т.е. в конечном итоге троек (4) и (5), А проделывает мысленно. В результате он пред-

лагает заказчику R^*, реализацией которой является Т-технология конструирования программы. У Z могут возникнуть два вопроса: во-первых, во всех ли ситуациях, интересующих Z, программа работает "правильно" (проблема непротиворечивости), во-вторых, все ли ситуации, интересные для Z, оказались учтенными (проблема полноты)? Предлагаем принять такую идеализацию - устранение проблемы непротиворечивости:

Относительно уже учтенных прецедентов программа работает "правильно" в том смысле, что она реализует именно эти прецеденты.

Заметим, что игнорирование проблемы непротиворечивости разумно лишь в данном случае, но не разумно, например, при обсуждении этой же задачи в коллективе разработчиков.

Итак, в ответ на R - эвристическое описание задачи Z - А предлагает нормативное описание R^* этой же самой задачи. R^* можно рассматривать как некоторую функцию, зависящую от А и от Z. Z может согласиться, что R^* удовлетворяет его целям, хотя возможна такая ситуация - а именно она и интересна для нас - что Z отвергает R^*. Но действуя так, Z не уподобляется оракулу, говорящему "да" или "нет", он выступает как источник новых прецедентов, причем, вводя их, Z объясняет, по какой причине он это делает, т.е. выдает новый принцип диагонализации (см. [I]). Возникает проблема - построить такую технологию конструирования программы, которая включала бы модель всего, в принципе бесконечного, диалога. Теоретический анализ такой ситуации можно проводить на различных уровнях: на абстрактном - в терминах А-пространств (см. [3]), на алгоритмическом - в терминах языков программирования, включающих в себя рефлексию, самоприменимость и т.д., где техническим аппаратом является теорема о неподвижной точке и т.п.

Резюмируя вышеизложенное, можно сказать, что диалог управляется тремя операторами: M_3 (он же Π_1), который по R дает Т; Π_2, который по Т дает R^*, и Ψ, который по R^* дает новое R (см. рис. I). Задача состоит не в том, чтобы накопить как можно больше прецедентов и принципов диагонализации, а в уточнении смыслов M_3, Ψ и Π_2 с той целью, чтобы искомая система, относительно которой можно надеяться, что Z ее примет, была бы замкнута относительно этих операторов, или (что то же самое) чтобы приемлемое R^* содержало модель всего бесконечного диалога.

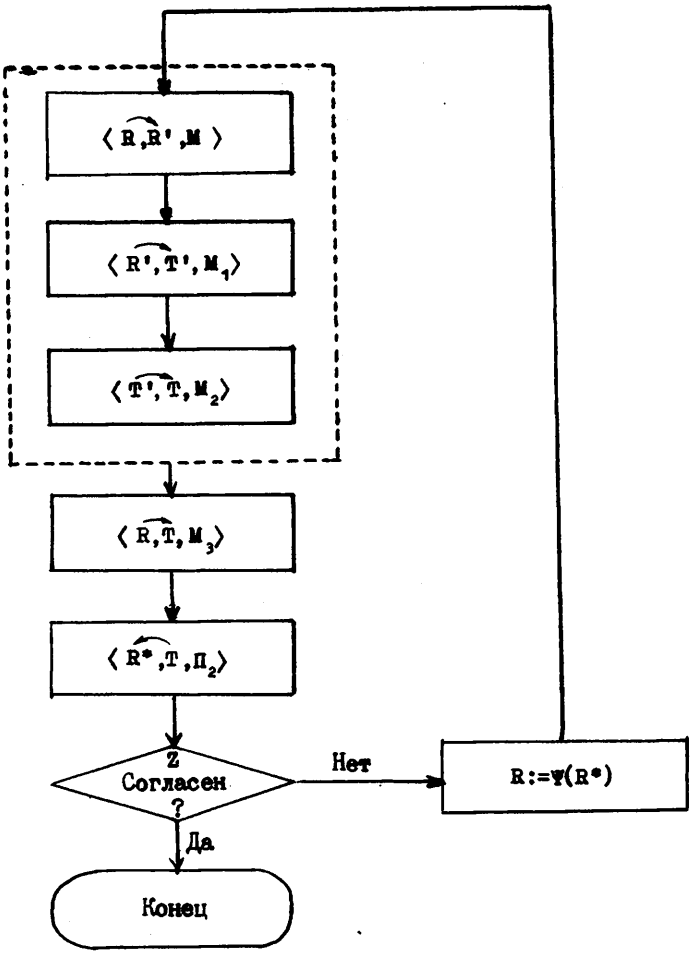

Схема диалога заказчика и архитектора.

§2. О моделировании бесконечного диалога

Постановка задачи организации общения заказчика и архитектора, в которую включалась бы модель диалога, описанного в §I, разумна хотя бы потому, что она является некоторой альтернативой этому наивному диалогу, продолжающемуся, в общем-то, неограниченно долго и прекращающемуся лишь в силу простого волеизъявления одного из его участников. В пользу того, что формализация подобной ситуации возможна, говорят следующие примеры из области математики.

I. Рекурсивно-перечислимое множество (отличное от множества всех натуральных чисел), элементы которого являются геделевскими номерами его перечислимых подмножеств (пример такого множества будет построен в конце этого параграфа).

2. Абстрактное пространство с заданным изоморфизмом между его точками и непрерывными функциями, например, A-пространство (см. [3]).

3. Нетрудно было бы представить себе такое формальное исчисление, выводимыми объектами которого были бы не только теоремы, но и его собственные подсистемы (и, возможно, что-нибудь еще). Насколько нам известно, такие исчисления в математической логике явным образом не исследовались, к ним естественно приводят рассмотрения, относящиеся к области искусственного интеллекта.

Для искомой формализации предположим, что имеется конкретный язык, в котором можно описать интересующую нас функциональную схему деятельности. Эта схема деятельности должна порождать подходящую систему прецедентов. Следует иметь в виду, что сами прецеденты могут быть весьма сложными и, в свою очередь, могут развертываться в систему более простых прецедентов - в этом смысле каждый нетривиальный прецедент представляет собой допустимую подсистему. Такое обстоятельство накладывает определенные требования на выразительные возможности языка. Для построения искомой модели должна быть выявлена достаточно полная система прецедентов. Разберемся в этом подробнее и уважем интуитивное требование полноты со структурой рассмотренного ранее диалога.

Возражения, исходящие от заказчика по поводу предложенного ему архитектором R^*, можно мыслить как функцию представлений Z о задаче, зависящую от предложений A. Можно представить Z поси-

телем нечеткой системы K (интуитивного представления о задаче), в которой он выделяет нечто относительно более четкое - $\bar{K}$ (приблизительная формализация задачи), причем $\bar{K}$ в ходе диалога с A может меняться и это изменение зависит от интуитивно мыслимого K и от предложений A . Задача заключается в создании такого $\bar{K}$, чтобы Z принял $\bar{K}$ в качестве адекватного заменителя K . В ответ на предъявленное R* заказчик, вообще говоря, выдает не только новые прецеденты, но и новые диагонализации. Можно представить дело так, что Z выбирает диагонализации из некоторой интуитивно мыслимой системы диагонализаций D , которая сама по себе необозрима и входит в K как составная часть. Конечно, в некоторых случаях можно считать, что Z обладает конечным множеством диагонализаций или даже одним оператором диагонализации, например, если Z является источником новых прецедентов и только, то его функционирование в этом качестве можно представить как наличие у Z одного оператора диагонализации. Рассмотрим, к примеру, иерархию формальных систем в языке формальной арифметики, получаемую последовательным присоединением утверждений о непротиворечивости (см. [4 , §4]). Каждое такое утверждение выступает в качестве нового прецедента, а алгоритм порождения этих прецедентов и есть оператор диагонализации. Но даже в связи с проблемами оснований арифметики возможны ситуации, когда порождаются новые принципы диагонализации, например, если не ограничиваться языком элементарной арифметики, а считать, что сам язык обогащается.

Вернемся к нашему случаю. R* в процессе диалога постоянно усложняется, могут появиться новые параметры и за счет этого - новые принципы диагонализации, которые управляют этими параметрами. Уточним, каким условиям должна удовлетворять готовая система. Будем считать, что если некоторое R*, возникшее в результате диалога, заказчиком не принимается, то R* все же включается в число допустимых прецедентов искомой системы (в силу устранения непротиворечивости). Будучи приближенной моделью искомой системы, R* развертывается в последовательность более простых прецедентов (т.е. допустимых подсистем R*), кроме того, R* замкнуто относительно ранее выявленных диагонализаций. В итоге критического обсуждения R* (когда порождается новое R* , быть может, тоже неприемлемое для Z) появляются новые прецеденты и новые диагонализации. Это происходит в результате применения операторов Ψ, M_3 и Π_2. Мы

хотим, чтобы готовая система (обозначим ее $\tilde{R}^*$) включала в себя все эти новые прецеденты и была замкнута относительно всех выявленных диагонализаций. С этой точки зрения можно охарактеризовать $\tilde{R}^*$ посредством двух операторов: $\theta_1(R_i,t)$ и $\theta(R_i,t)$, определенным образом зависящих от операторов Ψ, M_3, Π_2. Оператор θ_1 по каждому допустимому прецеденту R_i порождает последовательность новых прецедентов, заиндексированных параметром t, которые должны включаться в новую систему вместе с R_i, сюда входят и те прецеденты, которые содержатся в самом R_i. Аналогично θ задает последовательность диагонализаций, которые появляются в процессе обсуждения R_i и относительно которых должна быть замкнута вся система. Оператор θ_1 очевидным образом представляет собой некоторую диагонализацию; можно считать, что среди диагонализаций, порождаемых θ, имеется и та, которая задается посредством θ_1, т.е. оператор θ_1 поглощается оператором θ. Заметим, что операторы θ_1 и θ неявно зависят еще и от самой задачи, т.е. от K. Если представлять готовую систему, то естественно K заменить на $\bar{K}$. Это означает, в свою очередь, что оператор, описывающий условия замкнутости θ^K должен зависеть от геделевского номера искомой системы; если же фактически такая зависимость отсутствует, то задача решается совсем просто – это будет видно из дальнейших построений. Мы предлагаем решать задачу в более общем виде: требуется построить систему допустимых прецедентов, которая была бы замкнута относительно оператора $\theta^K(R_i,t)$, т.е. $(R_i \in \tilde{R}^*) \supset (\forall t \theta^K(R_i,t) \in \tilde{R}^*)$.

Предположим, что $\tilde{R}^*$ включает в себя опорные прецеденты, а так как θ^K определяется через операторы, управляющие диалогом, то в этом смысле $\tilde{R}^*$ является моделью этого бесконечного диалога.

Требуя замкнутость всей системы относительно всевозможных диагонализаций, мы тем самым полагаем, что сама система $\tilde{R}^*$ (или R^*) не является своей собственной подсистемой, а представляет собой нечто более сложное, чем каждая приближенная модель, входящая в нее, поскольку она претендует на роль готовой системы. Это хорошо согласуется с интуитивным пожеланием относительно того, чтобы Z отказывался диагонализировать результирующую систему, имея на то какие-то основания, а не просто путем волеизъявления. Если Z в результате наития будет выдвигать требования, на которые раньше не было и намека, то резонно представить это как замену задачи. Конечно, невозможно точно определить, где происходит замена, а где уточнение содержания задачи; поэтому мы не утверждаем, что Z

непременно сразу примет любую систему, замкнутую относительно всех диагонализаций, но все же разумно предположить, что в классе подобных систем, описанных в фиксированном языке, найдется такая система R^*, которой Z будет удовлетворен, т.е. условия замкнутости необходимы, но не достаточны для прекращения диалога.

В качестве первого приближения к реализации изложенной идеи предлагаем некоторое простое построение. Систему представляем в виде некоторого рекурсивно-перечислимого множества конструктивных объектов, которые отождествляем с их геделевскими номерами. Каждое рекурсивно-перечислимое множество M можно представить как область значений некоторой частично-рекурсивной функции h и геделевский номер h считать геделевским номером M. Итак, предположим, что имеется формальный алгоритмический язык, в котором описываются прецеденты, и что содержание операторов Ψ, Π_1, Π_2, а тем самым и оператора θ^K выяснено настолько, что они могут быть записаны в этом языке. Пусть a_0 - опорный прецедент; $\theta(e, b, t)$ - частично-рекурсивная функция (формализация оператора θ^K), где e - геделевский номер искомой системы; b - допустимый прецедент; $\lambda t\theta(e,b,t)$ - последовательность диагонализаций, порожденных прецедентом b. Для каждого e строим множество M_e, удовлетворяющее следующим условиям (! означает, что выражение определено):

1) $a_0 \in M_e$;

2) $(b \in M_e \;\&\; \theta(e,b,t)!) \supset (\theta(e,b,t) \in M_e)$;

3) $(b \in M_e \;\&\; \{b\}(t)!) \supset (\{b\}(t) \in M_e)$.

В силу условия 1), M_e - непусто, по условию 2) - замкнуто относительно оператора θ; в силу 3), каждый элемент M_e является геделевским номером его подмножества. Из построения M_e следует, что M_e рекурсивно-перечислимо и геделевский номер M_e можно построить эффективно по e, поэтому считаем, что имеется фиксированная рекурсивная функция f, которая по e дает геделевский номер M_e. С помощью теоремы о неподвижной точке строим e такое, что $\{e\}(x) = \{f(e)\}(x)$. Получаем, что множество с геделевским номером e, совпадающее с соответствующим множеством M_e, будет обладать нужными нам свойствами. Тем самым построена некоторая приблизительная модель интересующей нас системы.

С одной стороны, вышеизложенные соображения можно формализовать и развивать теорию подобных объектов независимо от реальности, на основании которой возникла теория, - такие рассмотрения

интересны и сами по себе в чисто математическом плане. С другой стороны, если имеем дело с ситуацией, где задействованы живые люди и решается практическая организация диалога, то необходимые описания (в том числе и оператора θ^K) будут производиться в полуформальном языке, проверка условий замкнутости осуществится "на глазок" и, вообще, контроль всего происходящего возможен лишь на "человеческом" уровне тем не менее учет подобных обстоятельств, стремление, чтобы они были как-то задействованы, представляется нам полезным и в этом случае. Мы надеемся, что преимущества описанного подхода будут проявляться по мере возрастания сложности систем, которые предстоит строить.

Литература

1. БЕЛЯКИН Н.В., ЖЕВЛАКОВА Е.А. Логико-семиотический подход к искусственному интеллекту. -Настоящий сборник, с. 34-41.

2. БЕЛЯКИН Н.В. Отношение теории к реальности. -В кн.: Эмпирическое предсказание и распознавание образов. (Вычислительные системы, вып. 79.) Новосибирск, 1979, с.7-11.

3. ЕРШОВ Ю.Л. Теория А-пространств. -Алгебра и логика, 1973, т. 12, № 4, с.369-416.

4. FEFERMAN S. Transfinite recursive progressions of axiomatic theories.- J.Symb.Logic, 1962, v.27, N 3,p.259-316.

Поступила в ред.-изд.отд.
17 октября 1980 года