УДК 007.51 + 004.055

А. И. Разумовский, канд. техн. наук, вед. инженер, e-mail: razumowsky@yandex.ru,
Институт проблем управления РАН, г. Москва

Информационная избыточность в разработке программных систем

Рассмотрена проблема разработки программных систем с точки зрения создания внешнего творческого контроля над информационным содержанием процесса разработки. Сделана попытка подчинить процесс создания программных систем избыточности информационного содержания посредством объединения управления как над алгоритмическими и структурными элементами, так и над любым возможным неформальным представлением данных, что способно обеспечить целостность разработки систем. Связующая роль между элементами проектирования при творческом контроле отведена эвристике доступности, которая действует как качественная оценка частоты или вероятности выбора за счет легкости воспоминания или ассоциации.

Ключевые слова: *проектирование программных систем, алгоритм, структуризация, эвристика доступности, творческая активность, информационная избыточность, творческий человеческий фактор*

Введение

При создании любой программной системы (ПС) определяющую роль играет комплексный мотив связности автоматической и неформальной составляющих среды разработки. В них системообразующее ядро всегда отведено нефор-мально действующему "человеческому фактору". Известно, что использование автоматизированных систем позволяет освободить разработчика от бремени согласования деталей, позволяя ему специально сосредоточиться на творческих аспектах процесса проектирования [1]. Одновременно с таким отношением существует отчетливо ясное представление автоматизированных систем как систем "накопления вычислительного знания о машинных методах теории регулирования и управления" [2]. То есть исходя из самого понятия автоматизации можно предположить, что универсум жизненного цикла автоматизированной системы обязан включать в себя элементы творчески значимого контроля. Более того, подтверждением этому является единственная возможность обеспечить ответственность за качество результатов и их оценочно конкретную достоверность. Как однозначно подчеркивается в работе [3], "проблема достоверности результатов решения задач, в значительной мере, лежит между реальным миром и теоретическими знаниями, и посредником между ними является человек". Действительно, противоречие между формальным и качественным представлением реальной задачи ведет к снижению содержательных характеристик программного продукта, что, в свою очередь, умеряет его надежность функционирования, а также комфорт использования.

Проблемы, возникающие при разработке и использовании ПС, так или иначе связаны с недостаточной поддержкой творческой активности человека, и в срезе имеющихся технологий проектирования ПС могут быть выражены следующим образом:

- неполнота информации в процессе проектирования и алгоритмизации;
- узость туннеля видимости результатов алгоритма;
- унификация и деперсонификация данных проекта.

Эти проблемы имеют две основные причины:

- рационализация процесса разработки ПС;
- унификация средств разработки ПС.

В работе [4] авторы сетуют: "Мы никогда не отыщем процесс, который дал бы нам возможность проектировать программы строго рациональным образом". Однако далее они отмечают: "Если мы держим в голове идеальный процесс, становится легче измерять успехи проекта", иначе говоря, творческие возможности проектировщика, считают авторы, должны быть подчинены определенным правилам процесса разработки ПС, что позволит обеспечить его управляемость. Таким образом, главным вопросом проектирования ПС остается проблема соотношения творческого и рационального в поиске, выработке

и принятии решений. Сужение области творчества разработчиков приводит к слабому профессиональному росту, безынициативности, небрежности [5]. Более того, в книге Р. Акоффа [6] выражается следующее мнение: "Ограничения, накладываемые общей логикой и абстрактной математикой на процедуры принятия решений, практически исключают возможность изучения *истинно творческих решений* и сводят науку о решениях к совокупности механических, а потому скучных и однообразных примеров принятия решений". Поскольку творческий процесс является спонтанным, необходимо учесть, что рожденная идея также может и не быть напрямую связана с решаемой проблемой [7]. Последнее означает, что при проектировании ПС необходимо заручиться возможностью зафиксировать все приходящие и имеющие информационно-техническое представление идеи. Фиксация рожденных идей, ассоциаций и оценок необходима также еще и из-за такой специфической особенности когнитивно-творческого аппарата человека, как ограниченный объем его кратковременной памяти — число Миллера: 7 +−2 [8]. В книге акад. Ларичева О. И. [9] делается предположение о двух основных путях решения человеком задач: "а) объединение отдельных единиц информации в блоки с целью одновременного восприятия этих блоков, б) использование специальных эвристик, приспосабливающих задачу к возможностям системы обработки информации человеком".

Математических моделей, адекватно конструктивно описывающих творческий человеческий фактор (ТЧФ), не существует. Конструктивно — значит, что математической формулой можно заменить определенную интеллектуальную функцию ТЧФ, или точнее, — его творческий импульс. Нет никакой надобности создавать, искать, отлаживать математические выражения, описывающие нечто, не являющееся ТЧФ или его деятельностью. Все подобные модели в ту или иную меру используют логико-вероятностный подход, основанный на субъективной экспертной оценке и, следовательно, далеко не вполне соответствуют действительности. Нами предлагается обустраивать ТЧФ индивидуально — тогда субъективность представления не только не будет помехой, но, наоборот, послужит важнейшей качественной опорой процесса участия ТЧФ в разработке. Как это организовать?

Построение избыточности информационного пространства

Общий порядок отдельного действия ТЧФ при разработке программных систем (ПС) таков: ТЧФ субъективно оценивает локальную задачу и творчески вырабатывает решение. Имеются два класса информации, обусловливающих выработку решений, — доступная информация и воображаемая информация, или перцептивное воображение [10]. Первая характеризуется всеми видимыми элементами процесса проектирования ПС: языковым и структурным представлением алгоритма, комментариями, отладочными данными, интерфейсными элементами среды разработки. Воображаемая информация — это то, что скрыто от глаз и рук проектировщика, но ассоциативно и интуитивно через множество инсайтов прокладывает дальнейший путь развития проекта ПС. С ценностной точки зрения воображаемая информация имеет перед видимой безусловное преимущество, однако именно воображаемая информация и испытывает наибольшие потери, поскольку ее невозможно впоследствии полностью воспроизвести, удастся лишь частично припомнить антураж творческого акта выработки решения.

Мы предполагаем, что воображаемую информацию можно частично зафиксировать в целях ее многократного повторного использования. Однако для реального использования такой информации придется программно регистрировать весь спектр возникающих в сознании данных, многие из которых совпадают по форме, отличаясь по содержанию. Это означает создание избыточного множества проектных элементов (ИМПЭ).

Итак, ИМПЭ — это зафиксированные и зарегистрированные факты и фрагменты процесса проектирования. Они включают в себя сопроводительные данные — примечания, фотографии, спецификации, требования, а также комплексные данные — произвольно (индивидуально значимо) объединенные в общем графическом контексте (на экране компьютера) информационные элементы проекта в сочетании с сопроводительными данными.

ИМПЭ способны обеспечить:

- совмещение в едином контексте проектной, аналитической и результатной информации;
- представление информации в реальном времени;
- реализацию в графических 2D- или 3D-средах.

Основное предназначение ИМПЭ — выработка и принятие проектных решений посредством опоры на эвристику доступности, которая действует как оценка частоты или вероятности выбора за счет легкости воспоминания или ассоциации. В качестве воспоминаний и ассоциаций выступают ИМПЭ, которые фиксируются и могут быть воспроизведены в последующий момент времени.

Аккумулированные вместе ИМПЭ позиционируются в едином плоскостном или пространственном контексте, который мы назовем креативно-контекстной формой (ККФ), поскольку определяющим фактором ее значимости будет концентрация и фиксация всех возможных ин-

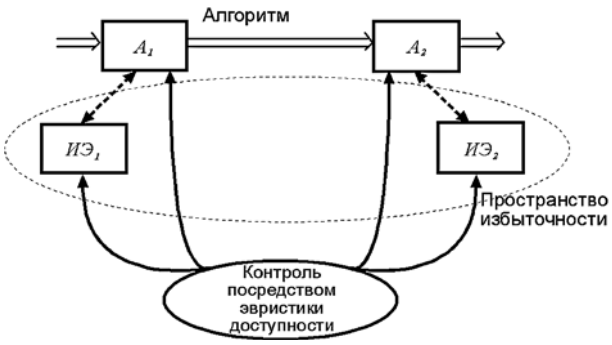

Рис. 1. Информационная модель восприятия и контроля информационной избыточности

формационных элементов в момент творческого озарения. Это своего рода фиксированные следы памяти — временные связи в коре мозга, служащие физиологической основой запоминания и воспроизведения. Кроме того, аккумуляция и фиксация ИМПЭ также сообразуется с теорией двойного кодирования А. Пайвио [11], согласно которой познание включает в себя деятельность двух отдельных подсистем: вербальной — обрабатывающей языковую информацию, и невербальной (образной), предназначенной для неязыковых объектов и событий.

Для выработки и принятия решения важно иметь в поле восприятия — на экране — входящие данные, аналитические данные (сравнения, уточнения, ошибки), динамику изменений определенных параметров, а также результаты, в любой возможной форме.

Рис. 1 демонстрирует информационную модель творческого восприятия ИМПЭ. У каждого алгоритмического элемента A_i существует избыточное множество образов (элементов) ИЭ, связанных с ментальным, формальным, ассоциативным представлением разработчика о нем. Кроме того, в это множество могут входить имеющие отношения к нему дополнительные вербальные или образные описания, а также данные промежуточных его состояний, например отладочная информация. При этом контроль связей внутри структуры алгоритма, соответствующий традиционному подходу алгоритмизации, сохраняется также.

Планирование и реализацию локального алгоритма намного проще проводить среди избыточного множества данных, где таковые всегда под рукой и всегда понятны разработчику, а следовательно, и комфортны для использования. Среди избыточных данных может быть теряемая информация, например, примечательная, иллюстративная или контекстно-отладочная, повторное восстановление которой затруднено, что, в свою очередь, тесно связано с понижением производительности творческого труда разработчика, а также с качеством его результата.

Пример реализации алгоритма на базе избыточности данных

Рассмотрим на предельно простом для наглядности примере влияние ИМПЭ на формирование тела алгоритма. Пример основывается на введении в состав упорядочиваемого множества элементов алгоритма определенно важных для разработчика дополнений, в общем случае создающих избыточное множество. Пример призван показать преимущества произвольного соотнесения ИМПЭ, что позволит при рассмотрении всего спектра элементов информационного пространства проектирования действовать индивидуально комфортно и вырабатывать решения творчески более активно, без необходимости создания специальных моделей или следования определенной технологии разработки.

Итак, пусть требуется разработать алгоритм определения свойств треугольника, если известны длины трех его сторон. Требуется найти, к какому виду треугольников — равностороннему, равнобедренному или произвольному — будет относиться исследуемый треугольник, заданный тремя целыми числами. Для простоты положим: дано пять отладочных троек длин сторон треугольников: (1,1,2), (2,1,3), (3,5,3), (2,2,2), (2,5,1).

Разобьем процесс разработки алгоритма на этапы, каждому из которых будет соответствовать добавление к программному коду хотя бы одного нового информационного элемента, в частности, оператора или поля данных. Первоначально нам нужно создать минимальный по содержанию алгоритм, который при этом позволит получить итоговый результат его функционирования. Этот результат, скорее всего, будет ошибочным, как и сам алгоритм, однако позволит поместить эти данные в контекст разработки на ККФ и осуществлять с помощью эвристики доступности наблюдение и выработку решений о развитии алгоритма.

Поэтапная разработка алгоритма показана на рис. 2.

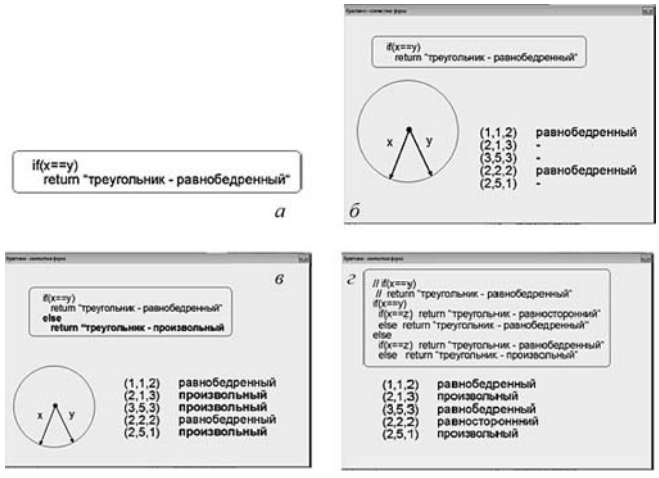

Рис. 2. ККФ последовательной разработки алгоритма

Первый этап создания алгоритма завершается реализацией всего одного оператора сравнения двух из трех входных величин (рис. 2, *а*). По результатам этого сравнения делается заключительный вывод об искомом свойстве треугольника. Экранная ККФ второго этапа заполняется программным кодом первого этапа, а также, для наглядности, рисунком, иллюстрирующим понятие "равнобедренный треугольник". Кроме того, осуществляется запуск скомпилированного алгоритма, и его результаты также располагаются на ККФ рядом с исходными данными и рисунком (рис. 2, *б*). Согласно содержанию ККФ, правильный результат получен лишь для 1-й тройки — (1, 1, 2).

Следующий этап создания алгоритма стимулируется необходимостью получения правильных результатов для остальных четырех отладочных троек. Это означает необходимость коррекции оператора, записанного на этапе 2, *а*. Добавим новый вывод при отрицательном ответе в операторе сравнения и поместим полученные после запуска скорректированной программы ее результаты на ККФ (рис. 2, *в*).

На очередном этапе разработки проводятся аналогичные действия. Кроме того, условие сравнения для третьего параметра при этом не удаляется, а комментируется неправильный код предыдущего этапа. Здесь следует обратить внимание, что в обстановке избыточности нет необходимости удалять лишние элементы — их всегда можно поместить на ККФ.

После запуска измененного алгоритма полученные результаты помещаем на очередную ККФ (рис. 2, *г*). Поскольку результаты удовлетворительны, можно считать процесс разработки завершенным.

Обращает на себя внимание тот факт, что накапливаемое множество избыточных информационных элементов не влияет на структуру и качество алгоритма непосредственно, поскольку вся необходимая информация в основном имеется в специализированных средствах отладки программ. Наличие ИМПЭ определяют возможность сохранения и последующего воспроизведения тех фактов, того индивидуального знания об объекте разработки, части которого стали результатом творческой выработки локальных решений и связаны между собой через эвристику доступности разработчика. Показанная на примере последовательность индивидуальных элементарных действий всегда присутствует в процессе создания ПС, однако почти вся информация, что участвует при выработке очередного локального решения, развивающего алгоритм, — отладочная, иллюстративная и т. п., — теряется, за ничтожным исключением. При обращении к структуре или коду алгоритма эти данные часто весьма трудно восстановить, а значит, для коррекции алгоритма будет необходимо вновь повторить некоторые шаги разработки, сличая результаты выполнения программы с ожидаемыми. На сохранение и последующее восстановление всей возможной информации о проекте, в том числе вербальной или образной, связанной между собой посредством эвристики доступности, и ориентирован настоящий способ разработки.

Обсуждение

В описаниях методологий и техник разработки ПС сегодня используются два подхода, которым условно можно дать именования "российский" и "западный". Для "российского" взгляда на процесс разработки ПС характерно стремление к формально-логическому описанию предмета. Для "западного", напротив, важно максимально приблизить комфорт разработки к пользователю, особенно это проявляется посредством эксплуатации принципа повторного использования программного кода, поскольку это "приводит к существенной экономии выразительных средств" [1, 12]. В "российском" подходе изъяном является упор на логику разработки, которую между тем часто нельзя сходу отследить и формализовать, а также на стремлении математически описать и обосновать личностные, нерациональные, случайные моменты, в том числе и возникающие в динамике процесса разработки ПС. Поэтому чтобы детально разобраться с причинами неудач, или наоборот, успехов проектов ПС, важно рассмотреть взаимосвязи разработчика, используемых им технологий и получаемого результата с точки зрения продуктивности и издержек.

На рис. 3 представлены пять уровней соответствия технологии и результата в разработке ПС. Нижний уровень характеризует готовую к расчету исходной задачи программу, результатом выполнения которой является целевое значение. Предшествующий исполнению ПС — уровень программирования, определяющий формально описанный на языке программирования алгоритм ПС. В рамках технологической поддержки решений на этом уровне выступают разнообразные среды разработки программного обеспечения, такие, например, как Microsoft Visual Studio.

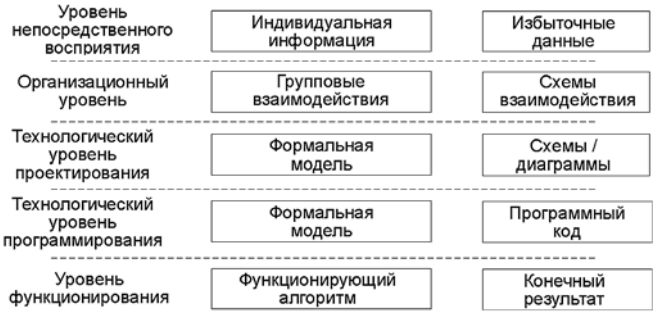

Рис. 3. Пять информационных уровня разработки ПС

Технологический уровень проектирования представляет собой комплекс мероприятий по поддержке целостности проекта в рамках его жизненного цикла. Среди средств обеспечения такой поддержки наиболее ярко выступают, во-первых, парадигмы проектирования, типа объектно-ориентированного проектирования (ООП), а во-вторых, технологии инструментальных средств разработки, например, система Rational Rose с использованием языка UML, или стандарты проектирования SADT/DFD. Кроме того, широко применяются также техники объединения проектных с программными решениями в виде различных шаблонов проектирования.

Организационный уровень ориентирован на человеческий фактор в плане результативности и качества производимых им действий и подкрепляется методологически созданием специальных условий для получения лучшего результата разработки. Здесь могут применяться разнообразные схемы и техники взаимодействия в социально-корпоративной среде — от экстремального программирования в смысле Agile [13] до парного программирования [14].

Наконец, верхний уровень — индивидуального восприятия — тот уровень, продуктивному исследованию которого посвящен этот текст. В литературе очень немного сказано о деятельности на этом уровне и в основном это общие слова, сетование на слишком малое внимание, обращаемое на человеческий фактор вообще и, в частности, в плане его восприятия и поведенческого, творческого комфорта. Можно сделать предположение, что причиной тому — объективная сложность, которая стоит за необходимостью учета всех нюансов деятельности человека как творчески активного индивида, вместо того чтобы рассматривать его как элемент рабочей схемы или методологии.

Таким образом, с уверенностью предполагая невозможность или неэффективность использования формально-математического или формально-информационного моделирования для исследования индивидуальных способностей создания и реализации качественных ПС, мы оказываемся в ситуации, когда лишь неформальный подход позволит выработать верное решение. Насколько невозможна точная математическая модель создания и реализации ПС, настолько невозможен численный эксперимент с точными проверяемыми результатами.

Человеческому фактору в разработке ПС уделяется чрезвычайно мало внимания, хотя, казалось бы, его ведущую роль и вклад в разработку ПС невозможно отрицать. Исключениями являются, например, работы [15, 16]. Так, в работе [16] автор, известный американский методолог и консультант разработки ПС А. Коуберн, выделяет несколько особенностей человеческой натуры, связанных с качеством разработки ПС: временная синхронизация в коммуникации между людьми; непостоянство людей (в нашем случае это перекликается с важностью индивидуальной выработки решения); инициативность, которая прямо влияет на успех проекта (в нашем случае — это ответственность за конкретный результат).

Надо при этом отметить, что ни в одной из книг и статей о влиянии человеческого фактора на разработку ПС не указывается творческий процесс, а лишь говорится о внешних его проявлениях, без соотнесения их в общем плане творческой инкубации и инсайта решения. Кроме того, внимание специалистов направлено больше на организацию коммуникации и внешней среды труда разработчиков, нежели на отдельные индивидуально значимые компоненты выработки решений.

На рис. 4 приведена упрощенная алгоритмическая схема операций при неформальном подходе к разработке ПС. Каждый элемент итерации выработки очередного решения начинается с его инкубации на основании представленных на ККФ ИМПЭ. После того как новое решение выработано и принято, оно оформляется в алгоритме на проектном или программном уровне и затем заносится и позиционируется на ККФ. В качестве данных, заполняющих ККФ, выступают наиболее значимые для разработчика элементы или их фрагменты. Это могут быть детали программного кода, комментарии, отладочные фрагменты, сопроводительные схемы и рисунки результата выполнения программы. Их ценность состоит не только в сохранении вообще, но и в соотнесении друг с другом определенным, важным для разработчика образом. Правильная совмещенность в избыточном контексте обеспе-

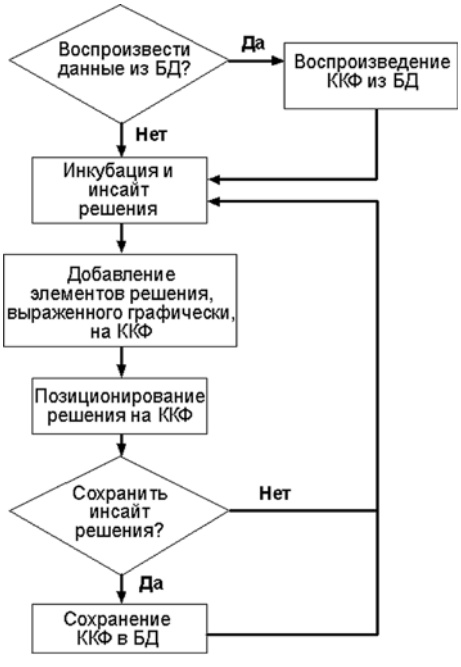

Рис. 4. Схема алгоритма действий творческого "контроллера"

чивает комфорт использования разработчиком эвристики доступности и, соответственно, инкубацию очередного решения.

Каждое решение может быть немедленно сохранено в базе данных ККФ или, при не полной заполненности ККФ, такое сохранение может быть отложено. Позиционирование решений на ККФ и их сохранение в базе данных позволит не только осуществить лучшую инкубацию последующего решения через эвристику доступности, но и воспроизвести воспоминание решения с сопутствующим антуражем. Это особенно важно при необходимости повторных возвратов к предыдущим итерациям разработки и выполняет отчасти роль расширенной кратковременной памяти, что непосредственно способствует лучшему обзору и использованию данных ККФ для новой инкубации и инсайта.

Практический образец разработки алгоритма

Вполне представить преимущества способа индивидуальной избыточности в разработке ПС, по-видимому, возможно лишь на реальном примере создания конкретного алгоритма.

Рассмотрим последовательность разработки алгоритма нахождения эквидистанты плоского контура (ЭПК). Особенность данного алгоритма заключается в необходимости хранения и использования всей возможной полноты информации о структуре исходного контура и деталях последовательного изменения алгоритма, в том числе такие данные, которые являются результатами локальных итераций проекта, а также разнообразные промежуточные расчеты. Необходимая целостность знаний об алгоритме может быть обеспечена только за счет создания такой информационно-функциональной среды, которая бы определяющим образом имела возможность накапливать, представлять и сохранять весь спектр информации.

Содержательная суть алгоритма расчета ЭПК заключается в последовательном нахождении значимых биссектрис углов между содержащими отрезки контура прямыми, а затем пересечении полученной сетки биссектрис эквидистантными к каждому из отрезков контура линиями. Полученные при последнем пересечении точки будут последовательно образовывать целевые эквидистантные контуры (рис. 5).

Проведем качественные оценки множеств информационных элементов, потребных для выработки очередного локального решения. Для хранения данных о каждом ребре исходного контура используется структура, состоящая из девяти полей, причем каждое из них применяется не только непосредственно для очередных расчетов, но и для контроля промежуточных результатов в текстовом и графическом виде.

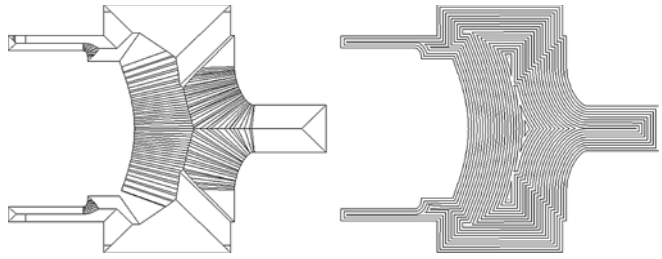

Рис. 5. Результаты выполнения алгоритма построения ЭПК

Сравним два случая сложности: в начале работы алгоритма и в середине. Ситуация начала работы алгоритма характеризуется в значительной мере лишь исходными данными, каждый из массивов полибиссектрисы (ПБ) содержит всего одну точку, исходную, поэтому наблюдение над данными для выработки решения представляет незначительные трудности. Второй случай отличается от первого насыщенными массивами ПБ, а также более сложным пониманием текущих результатов и их интерпретацией, без которой нельзя выработать и принять решение о возможностях дальнейшей разработки алгоритма.

Во втором случае число необходимых для наблюдения информационных элементов возрастает примерно от 1,5 до 7 раз для контуров с числом точек не более 500. Оценим, каково приблизительно число таких элементов для формирования условий инкубации очередного решения. Для первого случая число необходимых информационных элементов составляет примерно от 10 до 20. Для второго случая таких элементов требуется, соответственно, от 15 до 140. Увеличение в 1,5...7 раз требуемых для наблюдения элементов происходит из-за того, что число точек ПБ, координаты которых должны быть учтены, значительно возрастает от нуля в начале работы алгоритма. В простейшем случае имеем минимум 7 элементов — в начале работы алгоритма, и минимум 11 — в середине его работы, так как добавляются также еще две координаты и индекс отрезка пересечения. Но в среднем важных для наблюдения элементов увеличивается сразу на 4...5 точек, по паре координат с каждой стороны сегмента контура, что означает резкое возрастание элементов наблюдения и выработки решений примерно в 5 раз.

Естественно, такие оценки делаются исходя из индивидуальных представлений о важности информационного элемента в контексте прочих. Осуществляя наблюдение, поиск и принятие решения на базе эвристики доступности, разработчик выступает как эксперт, и сделанный им выбор опирается на личную эвристическую достоверность. В этом и состоит ценность творческого подхода, в противоположность компьютерному синтезу решений, где существует жесткая система правил выбора и предпринимаемых мер.

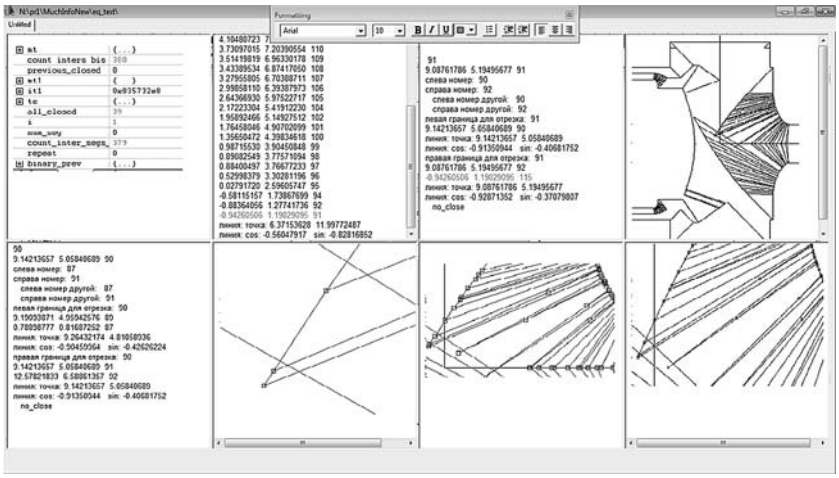

Рис. 6. ККФ разработки алгоритма ЭПК

Напротив, в условиях свободы творчества выработку решений и их оценку проводит сам разработчик на основе собственного опыта, знаний и способностей, применяемых вне технологических или аксиоматических препятствий.

Необходимость совмещения в едином контексте наблюдения наиболее важную с точки зрения разработчика информацию о некоторой проблемной ситуации, решение которой необходимо осуществить, иллюстрирует рис. 6. На этом рисунке изображена одна ККФ из полученных при проектировании алгоритма нахождения ЭПК в ситуации возникновения неожиданной ошибки на 25-м шаге итерации функционирования алгоритма.

Настоящая ККФ включает восемь слайдов для выбранных информационных элементов, каждый из которых содержит важную информацию для творческой выработки решений по исправлению возникшей ошибки функционирования. Данные четырех слайдов отображают ситуацию с графической точки зрения в разном масштабе. Еще три слайда содержат текстовую отладочную и log-информацию, относящуюся к текущим данным, связанную с пересечением ПБ. Последний слайд отображает сочтенные разработчиком важными отладочные данные, вырезанные как часть экрана в момент возникшей ошибки из среды MS Visual Studio.

Заключение

Подводя итог, перечислим главные цели, достигаемые использованием метода информационной избыточности:

- возможность непосредственно увидеть с помощью данных на ККФ новое расширение алгоритма;
- возможность в деталях восстановить информационное содержание любого этапа разработки;
- возможность вести одновременное наблюдение над несколькими вариантами алгоритма, и, как следствие, обеспечить качественное неформальное принятие лучших среди них решений;
- возможность наблюдать и искать решение в индивидуально значимом, конкретном контексте избыточных, совместно позиционируемых, данных.

Представленный в статье порядок организации накопления и воспроизведения ИМПЭ позволит соединить и зафиксировать в едином контексте любые графически выразимые информационные элементы разработки алгоритма ПС. Располагаемая на ККФ информация в своей совокупности содержит тот строй мыслей, который соответствует найденному решению в момент творческой инкубации и инсайта, свидетельствующему о верности решения.

Вообразим на мгновенье, что при разработке алгоритма ИМПЭ не применяются. В таком случае результатом станут многочисленные потери нюансов и атрибутов творческого инсайта, потери локальных данных проектирования и программирования, например, отладочных данных, издержки отсутствия многомерности представления информации и, наконец, потери от несогласования разноплановых данных между собой, т. е. целостности представления, что обеспечивается позиционированием разнообразных и избыточных элементов на ККФ. Все это вместе взятое потребует гораздо большего интеллектуально-творческого усилия, большего времени для нового "погружения" в содержательную суть проекта и деталей алгоритма. Это приведет к выработке и принятию менее надежных и качественных решений, а в некоторых сложных случаях, когда для их инкубации важно одновременно учесть достаточно много данных, количественно превышающих "магическое число" [8], найти правильный ответ станет и вовсе затруднено.

Представления о возможных преимуществах использования избыточности для построения качественных алгоритмов и систем своим истоком имеют книгу И. Пригожина и И. Стенгерс "Порядок из хаоса" [17]. В ней приведено множество аргументов в пользу избрания для разработки и поддержки работоспособности систем недетерминистского подхода, в частности, отсутствие возможности точного описания предмета. Приводимые в книге слова Л. Розенфельда как нельзя кстати подходят для подведения окончательной черты: "Включение спецификации условий наблюдения в описание явлений не произвольное решение, а необходимость, диктуемая самими законами протекания явлений и механизмом их наблюдения, что делает эти условия неотъемлемой частью объективного описания явлений" [17].

Список литературы

1. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.: Бином, 2001, 560 с.
2. **Васильев С. Н., Опарин Г. А., Феоктистов А. Г.** Интеллектный подход к автоматизации моделирования сложных управляемых систем // Современные проблемы прикладной математики и механики: Труды межд. конф. RDAMM-2001. Новосибирск: ИВТ СО РАН, 2001. Т. 6. Ч. 2. С. 159—168.
3. **Абрамова Н. А.** Логический подход к анализу достоверности идентификации // Проблемы управления, 2005, № 5. С. 77—82.
4. **Parnas D. and Clements P.** A Rational Design Process: How and Why to Fake It. IEEE Transactions on Software Engineering, 1986. Vol. 12, N. 2.
5. **Страуструп Б.** Язык программирования C++. М.: Бином, 2017. 1136 с.
6. **Акофф Р., Эмери Ф.** О целеустремленных системах. М.: ЛКИ, 2008. 272 с.
7. **Богоявленская Д. Б.** Психология творческих способностей. М.: Академия, 2002. 320 с.
8. **Miller G.** The Magical Number Seven, Plus or Minus Two: Some Limits on or Capacity for Processing Information, Psychological Review, USA, 1956, N 63. P. 81—97.
9. **Ларичев О. И.** Объективные модели и субъективные решения. М.: Наука, 1987. 144 с.
10. **Андерсон Дж. Р.** Когнитивная психология. СПб.: Питер, 2006. 589 с.
11. **Paivio A.** Imagery and verbal processes. New York: Holt, Rinehart, and Winston. 1971.
12. **Бек К.** Экстремальное программирование: разработка через тестирование. СПб.: Питер, 2017. 224 с.
13. **Cockburn Alistair.** Agile Software Development, 1st edition, Addison-Wesley Professional, December 2001.
14. **Cockburn A., Laurie W.** The Costs and Benefits of Pair Programming // The University of Utah, 2001. 11 p. URL: http://www.cs.utah.edu/~lwilliam/Papers/XPSardinia.PDF.
15. **Демарко Т., Листер Т.** Человеческий фактор: успешные проекты и команды. СПб.: Символ-Плюс, 2005. 249 с.
16. **Cockburn A.** Characterizing people as non-linear, first-order components in software development // Humans and Technology, October 1999.
17. **Пригожин И., Стенгерс И.** Порядок из хаоса: Новый диалог человека с природой. М.: Прогресс, 1986. 432 с.

A. I. Razumowsky, Ph. D., Leading Engineer, e-mail: razumowsky@yandex.ru,
Institute of Control Sciences RAS, Moscow, Russia

Information Redundancy in the Development of Software Systems

The problem of developing software systems from the point of view of organizing external creative control over the information content of the development process is considered. An attempt was made to subordinate the process of creating software systems for redundancy of information content, by combining control over both algorithmic and structural elements, and over any possible informal data representation, which will ensure the integrity of the development of software systems. The connectivity between design elements under creative control is the accessibility heuristic that acts as a qualitative estimate of the frequency or probability of choice due to the ease of recalling or association. The study of methodological levels of software system development is carried out. It was found that the individual level of creative activity of a person is almost not studied, therefore, the completeness and integrity of the development of software systems will not be achieved. The simplified scheme of actions of the developer is given at the informal approach with use of an excessive set of design elements. For positioning in a single visual context of redundant data, a creative-contextual form is proposed that serves to concentrate and fix all possible information elements at the time of creative insight. An example of the implementation of a geometric algorithm with the help of creative-contextual forms is considered, as well as estimates of the sets of information elements necessary for the creative development of the next local solution.

Keywords: *designing of software system, algorithm, structuring, accessibility heuristic, creative activity, information redundancy, creative human factor*

References

1. **Buch. G.** *Ob`yektno-oriyentirovannyy analiz i proyektirovaniye s primerami prilozheniy na C++,* Moscow, Binom, 2001, 560 p. (in Russian).
2. **Vasil'yev S. N., Oparin G. A., Feoktistov A. G.** Intellektnyy podkhod k avtomatizatsii modelirovaniya slozhnykh upravlyayemykh sistem, *Sovremennyye problemy prikladnoy matematiki i mekhaniki: Trudy mezhd. konf. RDAMM-2001.* Novosibirsk: IVT SO RAN, 2001, vol. 6, path. 2, pp. 159—168 (in Russian).
3. **Abramova N. A.** Logicheskiy podkhod k analizu dostovernosti identifikatsii, *Problemy upravleniya*, 2005, vol. 5, pp. 77—82 (in Russian).
4. **Parnas D. and Clements P.** 1986. A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering*, 1986, vol. 12, no. 2.
5. **Straustrup B.** *Yazyk programmirovaniya S++*, Moscow, Binom, 2017, 1136 p. (in Russian).
6. **Akoff R., Emeri F.** *O tseleustremlennykh sistemakh*. Moscow, LKI, 2008. 272 p. (in Russian).
7. **Bogoyavlenskaya D. B.** *Psikhologiya tvorcheskikh sposobnostey.* Moscow, Akademiya, 2002. 320 p. (in Russian)
8. **Miller G.** *The Magical Number Seven, Plus or Minus Two: Some Limits on or Capacity for Processing Information*, Psychological Review, USA, 1956, no. 63, pp. 81—97.
9. **Larichev O. I.** *Ob`yektivnyye modeli i sub`yektivnyye resheniya*. Moscow, Nauka, 1987. 144 p. (in Russian)
10. **Anderson Dzh. R.** *Kognitivnaya psikhologiya*. SPb.: Piter, 2006. 589 p. (in Russian).
11. **Paivio A.** *Imagery and verbal processes.* New York: Holt, Rinehart, and Winston. 1971.
12. **Bek K.** *Ekstremal'noye programmirovaniye: razrabotka cherez testirovaniye*. SPb.: Piter, 2017. 224 p. (in Russian).
13. **Cockburn Alistair,** *Agile Software Development*, 1st edition, December, Addison-Wesley Professional, December 2001.
14. **Cockburn A., Laurie W.** The Costs and Benefits of Pair Programming, *The University of Utah*, 2001. 11 p. URL: http://www.cs.utah.edu/~lwilliam/Papers/XPSardinia.PDF.
15. **Demarko T., Lister T.** *Chelovecheskiy faktor: uspeshnyye proyekty i komandy.* SPb.: Simvol-Plyus, 2005. 249 p. (in Russian)
16. **Cockburn A.** Characterizing people as non-linear, first-order components in software development, *Humans and Technology*, October 1999.
17. **Prigozhin I., Stengers I.** Poryadok iz khaosa: Novyy dialog cheloveka s prirodoy. Moscow, Progress, 1986, 432 p. (in Russian).