*Н. Г. Топольский, А. В. Крючков, И. М. Тетерин, Д. С. Грачёв,
К. А. Михайлов, До Хоанг Тхань (Россия, Вьетнам)*
(Академия ГПС МЧС России; e-mail: mihkir.94@mail.ru)

ОЦЕНКА УСТОЙЧИВОСТИ СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННЫХ ИНТЕГРИРОВАННЫХ СИСТЕМ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ ОБЪЕКТОВ

Рассмотрены вопросы устойчивости специального программного обеспечения автоматизированных интегрированных систем комплексной безопасности. Предложена математическая модель концептуальной схемы базы данных единичной программной системы. Описана техническая реализация данной модели на практике.

Ключевые слова: синтез программного обеспечения, автоматизированная система пожаровзрывобезопасности, базы данных, методы разработки специального программного обеспечения.

Основные вопросы автоматизации пожаровзрывобезопасности высокорисковых объектов заложены в [1], в том числе нефтеперерабатывающих производств в [2]. В связи с развитием различных информационных технологий существует необходимость анализа и оценки устойчивости применяемого на этих объектах *специального программного обеспечения (СПО)* [3].

Разработка структуры базы данных – один из главных этапов синтеза СПО, от успешности и точности действий на котором зависит итоговая устойчивость СПО. Согласно классическим правилам проектирования *баз данных (БД)* [4], фаза анализа и проектирования БД, работы по которой выполняет проектировщик (разработчик структуры БД) на одном из этапов жизненного цикла БД, включает в себя следующие процессы:

- формулирование и анализ требований;
- концептуальное проектирование;
- проектирование реализации;
- физическое проектирование.

Фаза реализации и функционирования БД включает в себя процессы:

- реализация БД;
- анализ функционирования и поддержка;
- модификация и адаптация.

Формулирование и анализ требований при синтезе СПО иногда воспринимаются преподавателями вузов как концептуальная фаза и фаза подготовки технического предложения [5]. Все иные процессы, перечисленные выше, составляют фазу проектирования.

При проектировании и разработке более сложных систем при условии, что заказчик и разработчик плохо разбираются в требованиях предметной области (для разработчика) и информационных технологий (для заказчика), процесс синтеза становится более запутанным. В этом случае разработчики начинают привлекать аналитиков, а заказчики – экспертов [6]. Уровень квали-

фикации, опыта и технической подготовки и тех, и других не всегда позволяет сторонам договориться. Это связано не только с тем, что стороны преследуют противоположные цели. Но и с тем, что им предстоит договариваться о том, какие части проекта будут реализованы в создаваемой подсистеме **автоматизированной системы пожаровзрывобезопасности (АСПВБ)**. А в процессе реализации выясняется, что часть требований, без которых невозможна реализация, всё равно не учитывается. "Правда заключается в том, что клиенты не знают, чего хотят. Обычно они не знают, на какие вопросы нужно дать ответ, и почти никогда не задумываются над задачей настолько детально, как это нужно указать в спецификации... Я пойду дальше и стану утверждать, что на практике клиенты, даже вместе с инженерами-программистами, не в состоянии указать полно, строго и корректно точные требования к современному программному продукту, прежде чем будут созданы и опробованы какие-либо версии продукта, спецификации к которому они составляют" [7]. Эта большая цитата подтверждает сказанное.

Итерационный процесс, который в технической литературе принято называть жизненным циклом **информационной системы (ИС)**, совершает свой очередной виток на очередном (более высоком) уровне, начиная от согласования требований и выработки спецификаций, до утверждения и согласования новых бюджетных расходов заказчика на подсистему АСПВБ. Это зачастую вызывает открытый конфликт руководства заказчика и разработчика, и, что самое важное, существенно влияет на её устойчивость. Это происходит в связи с тем, что детерминированные требования в **техническом задании (ТЗ)** при их реализации в СПО начинают приобретать вероятностный характер.

В случае, если разработчик представлен одним лицом и проект не очень велик, уместен подход, описанный в табл. 1 [6]. Как видно из описания этапов, они сильно отличаются от тех фаз, этапов и работ, которые описаны в нормативной базе^{1,2}. Разработка ТЗ с требованиями к **единой программной системе (ЕПС)** – только седьмой пункт из девяти предложенных, а первые шесть – предварительная работа, в том числе и по синтезу СПО. И самое интересное: ТЗ "будет представлять из себя БД после месяца работы". То есть схема БД, описанная после пяти месяцев работы и внедрения, станет основой для написания ТЗ, в котором будут изложены "требования к программе". Разумеется, в табл. 1 дан один виток спирали в спиральном методе разработки СПО. Немудрено, что после такого отношения большинства разработчиков (и заказчиков, которые принимают такие правила) к синтезу СПО, влияние на его устойчивость человеческого фактора будет подавляющим.

¹ ГОСТ 34.601-90. Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

² ГОСТ Р ИСО/МЭК 12207-99. Информационная технология. Процессы жизненного цикла программных средств

Таблица 1

Этапы разработки единичной программной системы
автоматизированной системы пожаровзрывобезопасности

№	Название	Время
1	Ознакомление с задачей	1 неделя
2	Выполнение пилот-проекта (муляж БД, дать представление о том, как будет работать итоговая версия)	1 неделя
3	Разработка первого варианта рабочего проекта (зависит от сложности задачи)	Примерно 2 месяца
4	Испытание (одному-двум сотрудникам будет поручена работа с БД, за это время будут сделаны необходимые изменения)	2 недели
5	Инсталляция клиентской части БД на все рабочие места, подключение к серверной части. Сотрудникам предоставляется выбор – работать по-старому или с БД	1 месяц
6	Внедрение	1 месяц
7	Разработка ТЗ (в ТЗ включается БД, сформированная после месяца работы с ней; изложены требования к программе)	2 недели
8	Работа согласно ТЗ	1 месяц
9	Сдача работы	1 неделя

Концептуальное проектирование при реальной работе над БД для конкретной ЕПС включает в себя опрос перспективного пользователя и формулирование в письменной форме списка возможных полей БД. Проектирование реализации включает в себя также выбор инструментальных средств для последующего физического проектирования. Наконец, физическое проектирование при реальной работе программиста над ЕПС заключается в написании кода на специальном языке или в работе с программой конфигурации таблиц БД, что совпадает с прописанной в учебниках стадией реализации БД (исключая синтез прикладных программ).

Реализация БД с помощью инструментальных средств *системы управления базой данных (СУБД)* до момента начала синтеза прикладных программ подразумевало описание её структуры в редакторе, позволявшем описывать поля БД их названия, тип, длину и другие свойства в зависимости от средства. Такой подход подразумевал наличие в голове разработчика чёткой концептуальной схемы БД ЕПС. Её собственно и формировали процессы с 1-го по 6-й в табл. 1.

Особо нужно подчеркнуть, что для двух разных разработчиков на языке одной СУБД она часто была различной даже для одного и того же ТЗ. Концептуальное проектирование структуры БД для любой ЕПС всегда сводится к диалогу между представителями заказчиков и разработчиков. По его результатам составляется некий промежуточный документ описательного характера, содержащий сведения о работе заказчика и о том, как заказчик хочет видеть результаты труда разработчика. Этот документ является прототипом ТЗ (п. 7 табл. 1).

Концептуальная схема БД ЕПС строится исходя из данных, которые предполагаемый пользователь будет вводить в БД со своего *автоматизированного рабочего места (АРМ)*, из введённых данных и правил преобразования первых во вторые. Исходя из этого, можно построить информационную формулу ЕПС.

Математическая модель концептуальной схемы БД ЕПС может быть построена на основе множеств. Пусть имеется некий список вводимых данных, перечень которых установлен на этапе формулирования требований к СПО:

$$S_{вх} = (a_{вх1}, a_{вх2}, \dots, a_{вхn}), \quad (1)$$

где $a_{вх1}, a_{вх2}, \dots, a_{вхn}$ – поля БД,

и список выходных данных, установленный на том же этапе:

$$S_{вых} = (a_{вых1}, a_{вых2}, \dots, a_{выхn}), \quad (2)$$

где $a_{вых1}, a_{вых2}, \dots, a_{выхn}$ – также поля БД.

$S_{вх}$ является подмножеством $S_{вых}$ в общем случае. Часть данных из $S_{вых}$ может быть получена путём расчётов и не войдет в $S_{вх}$, а часть – пополнит список $S_{вх}$. Обозначив список дополнений как

$$S_{дон} = (a_{дон1}, a_{дон2}, \dots, a_{донn}), \quad (3)$$

где $a_{дон1}, a_{дон2}, \dots, a_{донn}$ – дополнительные поля БД;

$$S_{расч} = (a_{расч1}, a_{расч2}, \dots, a_{расчn}), \quad (4)$$

где $a_{расч1}, a_{расч2}, \dots, a_{расчn}$ – расчётные поля БД, можем получить простую формулу

$$S_{вых} = S_{вх} \cup S_{расч} \cup S_{дон}. \quad (5)$$

Если исключить из (5) расчётные поля, то получим отсутствие транзитивности – требование, необходимое для того, чтобы БД находилась в (3) в нормальной форме. Поэтому окончательная информационная формула ЕПС, содержащая список полей, может быть представлена в виде

$$S_{вых} = S_{вх} \cup S_{дон}. \quad (6)$$

Впоследствии данный список может уточняться по мере разработки ЕПС. При этом $S_{вых}$ является множеством полей БД ЕПС.

Далее рассмотрим то, как это технически реализуется на практике.

После получения примерного списка полей БД абстрактный программист или проектировщик БД (администратор БД для больших систем) приступает к его анализу и построению логической структуры БД для всего СПО (в данном случае АСПВБ). На этом этапе выясняется то, какие данные и как необходимо хранить организационно (в одном или в нескольких файлах, на одном рабочем месте или на нескольких и т.п.). Целью логического проектирования является получение такой структуры БД (списка таблиц и полей в них), которая была бы наименее избыточной по отношению к хранящимся в ней данным.

Список с такой информацией может быть отсортирован по какому-то параметру или его значению. В общем виде он представляет собой многоуровневое дерево, сходное с графом иерархии требований на 4-м и 5-м уровнях его иерархии. Далее будем называть это дерево **деревом информационной схемы приложения (ДИСП)**, так как ЕПС и приложение для разработчика понятия одинаковые. Корнем его является элемент списка (номер условной "карточки"), первым уровнем – параметр, а следующим – значение этого параметра. Помимо этого, для значений, которые допускают выбор из нескольких значений (поля со списком или классификатором) в дереве появляется четвёртый уровень. Будем называть элемент списка (корень ДИСП) вместе с деревом значений по нему **информационной единицей хранения (ИЕХ)**.

После уточнения реляционной модели на стадии логического проектирования происходит её нормализация. Следует уточнить, что далее по тексту реляционная модель, информационная модель данных предметной области приложения в терминах CASE-технологии, информационная модель приложения, информационные схемы приложения будут употребляться для обозначения одного понятия – ДИСП. Следует отметить также, что абстрактными программистами или людьми, связанными с технологиями синтеза приложений при помощи ЯВУ (высокоуровневый язык программирования), термины "прикладная область" и "предметная область" используются для обозначения тождественных по сути понятий.

На стадии физического проектирования ответственный за реализацию сотрудник организации разработчика выполняет физическое описание полей по схеме: название, тип, длина, дополнительные данные о поле.

В "дополнительных данных о поле" может быть указано, например, число знаков после запятой. В зависимости от средства программирования (автоматизации или проектирования БД) и типов данных, им поддерживаемых или указанных в параметре "тип", параметры "длина" и "дополнительные данные о поле" могут меняться. Эти данные заносят в таблицу БД, представляющую собой текстовое описание в файле БД или отдельный файл. Далее на стадии реализации проекта СПО на инструментальном средстве указанная физическая структура БД может создаваться средствами системы программирования. Помимо собственно таблиц БД на этом этапе инструментальными средствами создаются описания индексных файлов и ограничения целостности данных. Могут выполняться и другие операции технологии клиент-сервер языковыми средствами той СУБД, при помощи которой БД создавалась.

Обычно корень многоуровневого дерева информационной схемы приложения реализуется с использованием одной большой таблицы БД, содержащей в качестве полей максимально возможную информацию об ИЕХ. Вспомогательные таблицы БД (файлы БД для локальных БД) используются чаще всего для определения классификаторов (списков, словарей, ограничений, перечислений), которые нужны для ввода значений в ИЕХ. В них заносится так называемая статическая информация, время изменения которой по сравнению со временем изменения информации в главной таблице во много раз ниже.

Примером может служить приложение с заказами, поставщиками и покупателями. "Свойства" покупателя и поставщика (данные, указанные в одной записи таблицы классификатора) заносятся один раз и много раз могут быть использованы в неизменном виде, в то время как требования на заказы (запись основной таблицы БД) добавляются или изменяются несколько раз в день (час, минуту и т.п.). Иногда интересы заказчика требуют наличия более одной основной таблицы. Но даже в этом случае можно точно показать, что в этом случае сам он может работать одновременно не более чем с одной таблицей. Поэтому возможно действовать, разделив информационную схему приложения на функционально независимые части, в которых одна таблица БД будет главной, а другие таблицы БД сыграют роль классификаторов (доменов).

Некоторые абстрактные программисты для удобства написания программ (снижения затрат на написание кода) используют один большой домен, записи которого имеют поле "номер словаря". По нему определяется, какой именно список будет использоваться для ввода или изменения данных в поля основной таблицы. Но такие технологические изыски могут быть скорее рецидивом, чем правилом.

Помимо реализации собственно логической структуры БД необходимо описание большого количества связей данной структуры с теми файлами, которые будут хранить необходимую информацию, а также настройка соответствующих профилей *ОС (Операционная система)*. Профили ОС используются для хранения данных о пользователе или приложении, им используемом. В них указываются такие данные, как имя пользователя, пароль, папки и файлы, используемые пользователем, папки, используемые для поиска основных и дополнительных файлов приложения, общее число открываемых файлов, размеры выделяемой памяти приложению и другие важные, но не имеющие отношения к ДИСП данные. Сходные метаданные могут храниться и в СУБД.

Всё это реализующий БД специалист должен аккуратно описывать каждым АРМ, время на такое описание составляет от 30 до 50 % общего времени создания БД.

При использовании подхода, основанного на *базовом классе задач автоматизации (БКЗА)*, время и трудозатраты на этом этапе могут быть сокращены при помощи создания инструментария, позволяющего большую часть данных операций выполнять автоматически. Появление хорошо отлаженного инструментария, написанного по понятным правилам, позволит также существенно повысить устойчивость создаваемого СПО. Реализация такого инструментария может отличаться для различных инструментальных средств. При этом его спецификации на различных платформах будут одинаковы. В этом случае возможна его разработка в качестве отдельного объекта СПО, устойчивость СПО будет значительно выше, чем при его синтезе традиционными методами.

Составной частью подобного инструментария является подход к предметным областям на основе БКЗА. В этом случае предусматривается, что данные любых предметных областей могут рассматриваться как составная большая таблица, содержащая условные "карточки" ИЕХ. При этом при необходимости такая таблица может быть дополнена деревом, содержащим определённые позиции группировки, что позволит разделить общую таблицу на удобные для обработки части. Это не только существенно сократит трудозатраты на реализацию БД, но и позволит повысить устойчивость СПО в АСПВБ за счёт гарантированно отлаженного и надёжно работающего инструментария.

Литература

1. *Топольский Н. Г.* Основы автоматизированных систем пожаровзрывобезопасности объектов. М.: МИПБ МВД России, 1997. 164 с.
2. *Абросимов А. А., Топольский Н. Г., Федоров А. В.* Автоматизированные системы пожаровзрывобезопасности нефтеперерабатывающих производств. М.: Академия ГПС МВД России, 2000. 239 с.
3. *Крючков А. В.* Обобщение опыта синтеза специального программного обеспечения на различных инструментальных средствах // Технологии техносферной безопасности. Вып. 3 (61). 2015. С. 252-263. <http://academygps.ru/ttb>.
4. *Тиори Т., Фрай Дж.* Проектирование структур баз данных. Т. 2, М.: Мир, 1982, 1984. 320 с.
5. *Избачков Ю. С., Петров В. Н.* Информационные системы: учеб. для вузов. СПб.: Питер, 2005. 655 с.
6. *Максименко Ю.* Корпоративная база данных: вопросы разработки и внедрения. <http://db-maker.narod.ru/indexrus.htm?page=databases.htm>.
7. *Чернявская Н., Чернявский Ю.* Требования к проекту. Классификация первый шаг к пониманию. http://www.comizdat.com/index_.php?id=478&in=kpp_articles_id.
8. *Вендров А. М.* Проектирование программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2005. 544 с.
9. Языки и средства спецификации программ // Требования и спецификации в разработке программ: сб. статей / пер. с англ. под ред. *В. Н. Агафонова*. М.: Мир, 1984. 346 с.
10. *Агафонов В. Н.* Спецификация программ понятийные средства и их организация, Новосибирск: Наука, 1987. 220 с.

Материал поступил в редакцию 11 ноября 2017 г.

Для цитирования: *Топольский Н. Г., Крючков А. В., Тетерин И. М., Грачёв Д. С., Михайлов К. А., До Хоанг Тхань.* Оценка устойчивости специального программного обеспечения автоматизированных интегрированных систем комплексной безопасности объектов // Технологии техносферной безопасности. – Вып. 2 (78). – 2018 – С. 55-62. DOI: 10.25257/TTS.2018.2.78. 55-62.

N. G. Topolsky, A. V. Kruchkov, I. M. Teterin, D. S. Grachev,
K. A. Mikhaylov, Do Khoang Tkhan (Russia, Vietnam)

ASSESSMENT OF THE SUSTAINABILITY OF THE SPECIAL SOFTWARE FOR COMPUTER-AIDED INTEGRATED COMPLEX SAFETY SYSTEMS OF OBJECTS

It deals with the issues of sustainability of special software for computer-aided integrated complex safety systems. The resulting sustainability depends on the success and accuracy of the steps during the design phase of the database structure. The study uses methods of system analysis, set theory. A mathematical model of the conceptual scheme of the database of single software system is proposed, its technical implementation in practice is described. The obtained results of the research will help the customer and the developer to design and develop single software systems for computer-aided integrated complex safety systems of objects. When using the approach based on the base class of task automation it will reduce the labor inputs for implementation of databases, and will also help to enhance the stability of the special software of the computer-aided integrated complex safety systems of objects.

Key words: software synthesis, computer-aided fire-explosion safety system, databases, special software development methods.

References

1. Topolsky N. G. *Osnovy avtomatizirovannykh sistem pozharovzryvobezopasnosti obektov* [Fundamentals of automated systems of fire and explosion safety of objects]. Moscow, Moscow Fire Safety Institute of Internal Affairs of Russia Publ., 1997, 167 p. (in Russian).
2. Abrosimov A. A., Topolsky N. G., Fedorov A. V. *Avtomatizirovannye sistemy pozharovzryvobezopasnosti neftepererabatyvayushchih proizvodstv* [Automated system of fire and explosion of oil refineries]. Moscow, Moscow Fire Safety Institute of Internal Affairs of Russia Publ., 2000, 239 p. (in Russian).
3. Kryuchkov A. V. Summarizing the experience of synthesis special software for different programming languages. *Tekhnologii tekhnosfernoj bezopasnosti // Technology of technosphere safety*, vol. 3 (61), 2015, pp. 252-263 (in Russian). Available at: <http://academygps.ru/ttb>.
4. Tiori T., Fraj Dzh. *Proektirovanie struktur baz dannyh* [Designing database structures]. Vol. 2, Moscow, Mir Publ., 1982, 1984, 320 p. (in Russian).
5. Izbachkov Yu. S., Petrov V. N. *Informacionnye sistemy* [Informational systems]. Saint-Petersburg, 2015, 655 p. (in Russian).
6. Maksimenko Yu. *Korporativnaya baza dannyh: voprosy razrabotki i vnedreniya* [Enterprise database: design and implementation]. Available at: <http://db-maker.narod.ru/indexrus.htm?page=databases.htm> (in Russian).
7. Chernyavskaya N., Chernyavskij Yu. *Trebovaniya k projektu. Klassifikaciya pervyj shag k ponimaniyu* [Requirements for the project. Classification is the first step to understanding]. Available at: http://www.comizdat.com/index_.php?id=478&in=kpp_articles_id (in Russian).
8. Vendrov A. M. *Proektirovanie programmogo obespecheniya ehkonomicheskikh informacionnykh sistem* [The software design of economical information systems]. Moscow: Finance and statistics, 2005, 544 p. (in Russian).
9. *Yazyki i sredstva specifikacii program* [Programming languages and program specification tools]. *Trebovaniya i spetsifikatsii v razrabotke program*, trans. from English, ed. by V. N. Agafonov, Moscow, Mir Publ., 1984, 344 p. (in Russian).
10. Agafonov V. N. *Specifikaciya programm ponyatijnye sredstva i ih organizaciya* [Specification of programs conceptual tools and their organization]. Novosibirsk, Nauka Publ., 1987, 220 p.

For citation: Topolsky N. G., Kruchkov A. V., Teterin I. M., Grachev D. S., Mikhaylov K. A., Do Khoang Tkhan. Assessment of the sustainability of the special software for computer-aided integrated complex safety systems of objects. *Tekhnologii tekhnosfernoj bezopasnosti // Technology of technosphere safety*, vol. 2 (78), 2018, pp. 55-62 (in Russian). DOI: 10.25257/TTS.2018.2.78.55-62.