УДК 004.056.2

Е.В. Архипочкин

ОСОБЕННОСТИ И СВОЙСТВА РЕЗИДЕНТНОГО КОМПОНЕНТА БЕЗОПАСНОСТИ В АСПЕКТЕ ОБЕСПЕЧЕНИЯ ЦЕЛОСТНОСТИ АИС

Основной задачей исследования является построение и обоснование состава и алгоритмов работы системы обеспечения целостности (СОЦ), защищающей данные, хранящиеся и обрабатываемые в базе данных (БД), а также программные компоненты серверной части уровня БД от атак со стороны нарушителя, обладающего правами администратора СУБД, на основе которой функционирует автоматизированная информационная система (АИС). Опишем модель нарушителя.

1. Нарушитель имеет возможность подключиться к БД АИС на низком уровне.
2. На низком уровне нарушитель имеет возможность выполнять просмотр, изменение, удаление записей таблиц, самих таблиц, функций, методов, пакетов, триггеров БД, запуск любых методов, остановку пакетных заданий сервера СУБД (jobs), изменение времени и параметров их запусков.
3. Следствие из п. 2. Нарушителю известен открытый код компонентов АИС серверной части уровня СУБД. Он может исследовать исходный код всех хранимых серверных процедур, функций, пакетов, триггеров.
4. Нарушитель не может получать открытую информацию из канала связи клиент-сервер.
5. Нарушитель не может исследовать функционирование серверной части АИС уровня ОС – заниматься отладкой как программных компонентов АИС уровня ОС (библиотек), так и отладкой серверной части используемой СУБД с целью получения информации, передающейся внутри сервера СУБД в процессе функционирования системы.
6. Цель нарушителя состоит в том, чтобы несанкционированно измененные или внесенные им данные обрабатывались и предоставлялись системой как целостные.
7. Нарушитель не заинтересован в нарушении работоспособности системы.

Особенностью СУБД является то, что на программные компоненты подуровня представления информации СУБД хранятся в БД в специальных таблицах словаря данных. Их исходный код в открытом виде доступен любому пользователю СУБД, имеющему соответствующие права.

Производители СУБД предлагают различные средства для сокрытия выполняемого исходного кода (например, утилита WRAP в СУБД Oracle), но эффективного решения не существует. Так, для всех современных версий СУБД Oracle в свободном доступе существует утилита REWRAP, переводящая код, закрытый утилитой WRAP, в исходный. По этой причине в модели нарушителя особо отмечен пункт 3 – исходный код всех программных компонентов СУБД доступен для просмотра и модификации пользователем с правами администратора СУБД.

Предложенная модель нарушителя является актуальной в силу объективных условий возможности существования реального нарушителя, действующего в рамках предложенной модели. Она основывается на типовой архитектуре АИС, имеющей в своем составе СУБД. Нарушитель, соответствующий предложенной модели, способен реализовать угрозу нарушения целостности.

Одной из наиболее часто используемых моделей защиты информации в АИС является субъектно-ориентированная модель (СО-модель) и связанная с ней концепция изолированной программной среды (ИПС). Изолированная программная среда это совокупность (множество) программ, в которой:

- никакая активизированная программа не влияет на другую активизированную программу;
- никакая активизированная программа не влияет на данные, которые используются для создания (активизации) другой программы;
- каждая программа может использовать только те данные, которые ей разрешено использовать политикой безопасности;
- каждая программа может активизировать только те программы, целостность которых установлена и активизация которых ей разрешена политикой безопасности.

В силу архитектуры рассматриваемой АИС и сформулированной модели нарушителя полное выполнение требований базовой теоремы о генерации ИПС является невозможным, поскольку в роли «программы» может выступать либо используемый нарушителем программный продукт, не являющийся частью АИС, либо модифицированный нарушителем компонент АИС.

Решение задачи может быть достигнуто расширением ИПС до доверенной вычислительной среды (ДВС), согласно модели, наиболее полно представленной в исследовании Конявского В.А. в приложении к защите электронного документооборота [1]. ДВС принадлежит классу СО-моделей и позволяет выполнить более точный учет особенностей АИС типовой архитектуры и модели нарушителя.

ДВС представляет собой фрагмент среды взаимодействия компонентов АИС, для которого установлена и поддерживается в течение заданного интервала времени целостность объектов, целостность взаимосвязей между ними и выполняемых функций.

Концептуальные отличия модели ДВС от модели ИПС перечислены ниже.

1. Требование неизменности программных средств в модели ИПС заменяется требованием целостности компонентов АИС, из которого вытекает требование сохранения в течение рассматриваемого периода времени в требуемом диапазоне состава объектов и процессов, их взаимосвязей и параметров функционирования.

Вычислительная среда меняется во времени, и задача защиты информации – обеспечение изменений в предусмотренном штатном режиме. Но тогда задача становится динамической, а динамический характер информации с необходимостью требует отслеживания ее изменений во времени. Если ставится задача контроля целостности системы, то необходимо реализовать измерение состояния меняющихся во времени параметров системы и сравнение измеренных значений с эталоном.

2. Доверенная вычислительная среда принципиально предполагает наличие некого «контролера», отслеживающего ее параметры — компонента безопасности.

Если компонент безопасности будет находиться «вне» среды, то обязательно должен быть некий элемент, разрешающий такой контроль и вмешательство в процесс функционирования среды при фиксации нежелательных режимов. В таком случае задача сводится к предыдущей. Следовательно, компонент безопасности необходимо встраивать в состав АИС, это должен быть резидентный объект. В соответствии с полученным заключением далее будем его называть резидентным компонентом безопасности (РКБ).

С практической точки зрения, включение РКБ в состав АИС предоставляет широкие возможности для возложения на этот компонент ряда дополнительных,

помимо контроля состояния системы, функций: оповещения, регулирования и управления процессами в АИС.

3. С одной стороны, РКБ предназначен для защиты АИС, а с другой, — сам является частью АИС и, таким образом, сам нуждается в защите.

Теоретически идеальная защита вычислительной среды недостижима, но практически сформулированный тезис сводится к тезису о том, что сам РКБ должен характеризоваться намного более высоким уровнем защищенности, нежели остальные компоненты АИС. Поэтому при практической реализации целостность РКБ должна обеспечиваться более высоким уровнем (для уровня представления СУБД — уровнем ОС; для уровня ОС — аппаратными методами).

Учитывая, что нарушитель согласно модели имеет возможность модифицировать как данные, так и программные компоненты АИС уровня БД, с практической точки зрения задача сводится к созданию и поддержанию доверенной вычислительной среды, а не к теоретическому доказательству и практической реализации ее изолированности.

РКБ должен участвовать в обеспечении целостности объектов на всех этапах функционирования режима эксплуатации АИС. Формирование доверенной программной среды может рассматриваться как важнейший элемент решения обеспечения доверия к системе в целом.

Могут быть выделены следующие особенности и свойства РКБ как ключевого компонента системы обеспечения целостности:

1. Автономность. РКБ должен быть максимально автономен относительно компонентов АИС, безопасность которых он должен обеспечивать.

2. Примитивность. РКБ по своей функциональной сущности должен представляться выделенным особо защищенным компонентом, выполняющим ограниченный набор функций.

3. Управляемость. Сигналы на изменение набора правил обеспечения целостности должны поступать РКБ извне по отношению к самому РКБ и защищаемым компонентам АИС. Подлинность и корректность таких команд должны устанавливаться с использованием криптографических методов.

4. Размещение. Система имеет линейную архитектуру (линейная система), если множество объектов строго упорядочено – всякому объекту, за исключением начального, предшествует один и только один объект. Иначе, любой процесс технологии инициируется одним и только одним процессом.

$O = \{o_i,\ i = \overline{1,N}\}$ система из N взаимосвязанных объектов o_i, (технология O, состоящая из N процессов o_i).

Система с линейной архитектурой может быть отображена в виде последовательности объектов o_i, начинающейся с объекта o_1 и заканчивающейся объектом o_N.

Определение 1. Линейная система является целостной, если установлена целостность каждого из ее объектов $o_i \in O,\ i = \overline{1,N}$.

Определение 2. Связь между любой парой объектов (o_i, o_{i+1}) при проверке целостности линейной системы описывается (n_i+1)-местной функцией проверки целостности объекта:

$$f_i = f_i(o_i) = f_i(o_i, p_{i,1}, p_{i,2}, ..., p_{i,n_i}) = o_{i+1},\quad i = \overline{1, N-1},$$

где $p_{i,1}, p_{i,2},\ldots, p_{i,n}$ — параметры объекта o_{i+1}. Функция f_i, $i=\overline{1,N-1}$, устанавливает целостность объекта o_{i+1}, если целостность объекта o_i зафиксирована. При этом функции $f_i(o_i)=o_{i+1}$, $i=\overline{1,N-1}$, являются функциями следования.

Справедливо

Утверждение 1. Целостность линейной системы установлена тогда и только тогда, когда установлена целостность объекта o_N.

Следовательно, задача установления целостности линейной системы эквивалентна задаче установления целостности объекта o_N.

Пусть подмножество $M \subset O$ есть множество объектов системы, целостность которых установлена.

Определение 3. Множество M разрешимо, если существует алгоритм A_M, который по любому объекту o_i дает ответ, принадлежит o_i множеству M или не принадлежит.

Можно использовать эквивалентное (в соответствии с тезисом Черча) определение

Определение 3.(1). Множество M разрешимо, если оно обладаем вычислимой всюду определенной (общерекурсивной) функцией χ_M, такой, что

$$\chi_M(o_i)=\begin{cases}1, \text{если } o_i \in M\\ 0, \text{если } o_i \notin M\end{cases} \qquad (1)$$

Определение 4. Множество M называется перечислимым, если оно является областью значений некоторой общерекурсивной функции, т.е. существует общерекурсивная функция $\Psi_M(x)$ такая, что $o_i \in M$, если и только если для некоторого $x \in N$ $o_i=\Psi_M(x)$. Функция $\Psi_M(x)$ называется перечисляющей для множества M.

Классическим результатом теории алгоритмов является следующее утверждение.

Утверждение 2. О разрешимости.

Множество M разрешимо, если и только если M и $\overline{M}$ перечислимы.

Определение 5. Линейная система является целостной, если M — разрешимое множество и M совпадает с O: $M = O$.

В исследовании [1] доказываются следующие два утверждения.

Утверждение 3. Об использовании РКБ.

Задача контроля целостности системы разрешима только при использовании РКБ.

Следствие 1. Установление целостности возможно только при расширении АИС РКБ.

Следствие 2. Установление целостности АИС невозможно только за счет средств одного уровня без использования РКБ.

Рассмотрим теперь вопрос о размещении РКБ в системе, описываемой линейной структурой.

Будем полагать, что РКБ внедрен в систему как объект o_0.

Утверждение 4. О размещении РКБ в линейной системе.

РКБ может быть размещен в системе линейной структуры произвольным образом, при условии, что в системе присутствует объект o_0.

Таким образом, установлено, что РКБ (по крайней мере, его часть) должен располагаться на уровне, целостность которого считается гарантированной. В рамках принятой модели нарушителя, таким уровнем является серверная часть АИС уровня ОС.

5. Структура. Первый шаг алгоритма ступенчатого контроля для создания ДВС должен базироваться на применении некоторой неизменяемой (немодифицируемой) процедуры, выполняющей роль доверенной точки для развертывания следующих этапов.

Учитывая типовую архитектуру АИС, модель нарушителя и определенный выше уровень с гарантированной целостностью, устанавливаем, что такой стартовой точкой или доверенным рубежом могут являться:

- Серверная библиотека уровня ОС, написанная на языке C++ или на Ассемблере. В этом случае необходимым условием безопасности является недоступность такой библиотеки для дизассемблирования или отладки силами нарушителя.
- Аппаратная часть АИС, представляющая собой, например, накопитель на флеш памяти с USB интерфейсом с неизменяемой доверенной информацией, или устройство типа *USB-CryptoKey* и т. п.

Исходя из этого, можно сделать заключение, что РКБ должен размещаться на всех уровнях в диапазоне от доверенного рубежа, до уровня, которому принадлежат защищаемые объекты включительно. Следовательно, РКБ сам по себе является не отдельным программным или аппаратным модулем, но состоит из набора модулей, каждый из которых принадлежит своему уровню из определенного выше диапазона. Доверенным рубежом в случае АИС, построенной на основе СУБД может являться один из следующих модулей:

- Серверная библиотека уровня ОС, написанная на языке C++ или Ассемблер. В этом случае необходимым условием безопасности является недоступность такой библиотеки для дезассемблирования или отладки силами злоумышленника.
- Аппаратная часть АИС, представляющая собой, например, USB-драйв с неизменяемой доверенной информацией, устройство типа USB-CryptoKey и т. п. В этом случае условием является недоступность злоумышленнику информации, хранящейся на таком устройстве, а также невозможность ее несанкционированного изменения.

В работе [1] формально показано, что с точки зрения защищенности более предпочтительным является вариант аппаратной реализации части алгоритмических процедур, а также размещение последнего рубежа в отдельном устройстве.

Исходя из этого, можно сделать заключение, что РКБ должен размещаться на всех уровнях в диапазоне от доверенного рубежа до уровня, которому принадлежат защищаемые объекты включительно. Следовательно, РКБ сам по себе является не отдельным программным или аппаратным модулем, но состоит из набора модулей, каждый из которых принадлежит своему уровню из диапазона. Рассмотрим как более общий вариант, когда доверенным рубежом считается аппаратный модуль.

В этом случае РКБ должен иметь структуру, показанную в табл. 1. Таким образом, РКБ должен состоять из взаимодействующих модулей, размещенных на разных уровнях.

В статье [2] показана реализация системы обеспечения целостности, ядром которой является РКБ с установленными в настоящей работе свойствами. В практическом аспекте РКБ является монитором безопасности системы. Там же доказывается следующее утверждение.

Таблица 1

Уровень	Модуль РКБ	Целостность
Аппаратный (USB-драйв, USB-CryptoKey)	Мастер-ключ, образ или значение хэш-функции модуля следующего уровня	Декларируется целостным и не скомпрометированным
Уровень ОС серверной части	Библиотека серверной части АИС уровня ОС	Целостность устанавливается на основе информации, хранящейся на аппаратном уровне
Уровень БД	Хранимый программный компонент уровня БД (пакет). Таблица, содержащая правила обеспечения целостности	Целостность устанавливается на основе информации, хранящейся на аппаратном уровне с помощью модуля уровня ОС

Утверждение 5. Атака, целью которой является несанкционированное изменение обрабатываемых АИС данных, считается успешной тогда и только тогда, когда система предоставляет измененные данные пользователю на высоком уровне (либо через API АИС).

Покажем, что свойства РКБ в сочетании с предлагаемым взаимодействим компонентов СОЦ делает систему с внедренной СОЦ устойчивой к атакам со стороны нарушителя, действующего согласно принятой модели.

Введем следующие обозначения:

val(TAB.d) или просто *val(d)* – значение поля *d*, хранящееся в таблице БД *TAB* в строке со значением первичного ключа *pk*;

$val_u(TAB.d)$ или просто $val_u(d)$ – значение, переданное пользователю АИС, как хранящееся в таблице *TAB* в строке со значением первичного ключа *pk*.

*null** – не предоставление информации и/или не функционирование клиентской части АИС в целом и/или информирование ответственных лиц о нарушении правил ограничения целостности в системе;

job – состояние, в котором находится АИС при работающем мониторе безопасности;

$\neg job$ – состояние, в котором находится АИС при неработающем мониторе безопасности.

Тогда работу клиентской части АИС в плане передачи информации из БД пользователю АИС можно представить как преобразование:

$$Client\ (val(d)) = val_u(d).$$

При внедрении СОЦ в АИС преобразование *Client* заменяется на преобразование $Client^{MB}$, такое что:

$$Client^{MB}(val(d)) = \begin{cases} val_u(d), \ если\ job; \\ null^*, если\ \neg job. \end{cases}$$

Аналогично работу API серверной части АИС можно представить как преобразование:

$$API(val(d)) = val(d).$$

Функционирование подсистемы обеспечения горизонтальной целостности таблиц при защите таблицы TAB и включение столбца d в перечень защищенных может быть представлено как:

$$MC(val(d)) = \begin{cases} состояние\ строки\ mc,\ если\ строка\ целостная; \\ состояние\ строки\ \neg mc,\ в\ противном\ случае. \end{cases}$$

Обозначим API_{MC} пакет API с внедренной проверкой горизонтальной целостности возвращаемых значений. Тогда:

$$API_{MC}(val(d)) = \begin{cases} val(d),\ если\ строка\ в\ состоянии\ mc; \\ null^{*},\ если\ строка\ в\ состоянии\ \neg mc. \end{cases}$$

Обозначим атаку, целью которой является несанкционированное изменение защищенного значения d таблицы TAB в некоторой строке, как:

$$A1(d) : val(d) \rightarrow val'(d).$$

Нарушитель, действующий в рамках модели, способен провести атаку *A1(d)* на строку, отбираемой по значению первичного ключа pk, введя команду:

update TAB set d=<новое значение val'(d)> where pk=val(pk);
commit;

При этом пользователь получит информацию $Client(val'(d)) = val'_u(d)$. Объектом атаки *A1* являются хранящиеся в системе данные.

Обозначим атаку, целью которой является несанкционированное изменение кода критичного программного объекта БД $p \in P^{*}$, как:

$$A2(p) : p \rightarrow p'.$$

В практическом аспекте это означает введение злоумышленником команды:

create or replace package body p_api as
...;

Объектом атаки *A2* являются технологии обработки данных.

Существенным теоретическим результатом является:

Утверждение 6. Для обеспечения устойчивости АИС к атакам A1 и A2 необходимо и достаточно передавать пользователю информацию по следующему принципу:

$$val_u(d) = Client^{MB}(API_{MC}(val(d))).$$

Доказательство.

Необходимость. Пусть в АИС не внедрена СОЦ. Тогда пользователь получает информацию как $val_u(d)=Client(val(d))$ или как $val_u(d)=Client(API(val(d)))$.

В случае использования метода передачи информации $val_u(d)=Client(val(d))$, нарушитель, действующий в рамках модели, способен провести атаку $A1(d) : val(d) \rightarrow val'(d)$.

При этом пользователь получит информацию $Client(val'(d)) = val'_u(d)$. Таким образом, атака $A1$ на хранящиеся данные будет успешной.

В случае использования метода передачи информации $val_u(d)=Client(API(val(d)))$ помимо возможности успешного проведения атаки $A1$, нарушитель может провести атаку $A2$:

$$A2(API) : API \rightarrow API', \text{ так, что } API'(val(d)) = val'(d),$$

то есть изменить код некоторого пакета API так, чтобы его методы, получая корректное значение $val(d)$, вместо него передавали измененное значение $val'(d)$. Атака $A2$ будет успешной.

Достаточность. Пусть в систему внедрена СОЦ. Тогда пользователь получает информацию по принципу: $val_u(d) = Client^{MB}(API_{MC}(val(d)))$.

Пусть нарушитель проводит атаку $A1(d) : val(d) \rightarrow val'(d)$.

$API_{MC}(val'(d))=null*$, следовательно $Client^{MB}(API_{MC}(val'(d))) = null*$.

Согласно утверждения 5 атака не успешна.

Пусть нарушитель проводит атаку

$$A2(API) : API \rightarrow API', \text{ так, что } API'(val(d)) = val'(d).$$

Из свойства динамического аспекта монитора безопасности следует, что при своем запуске монитор безопасности производит проверку целостности каждого $p \in P*$ и блокирование p в эксклюзивном режиме. В случае обнаружения нарушения целостности любого пакета монитор безопасности не запускается, следовательно, система переходит в состояние $\neg job$. Изменение заблокированного монитором безопасности пакета p возможно только в случае уничтожения блокирующей его сессии. В этом случае система также перейдет в состояние $\neg job$. Известно, что $API_{MC} \subset P*$. Следовательно, в результате атаки А2 система перейдет в состояние $\neg job$, не зависимо от того, произошла атака А2 при включенном мониторе безопасности или при выключенном.

Следовательно, $Client^{MB}(API'_{MC}(val(d)))=null*$. Согласно утверждения 5, атака не успешна.

Не трудно видеть, что комбинация атак А1 и А2 будет также не успешной:

$$Client^{MB}(API'_{MC}(val'(d)))=null*.$$

Доказательство завершено.

Утверждение 7. Атака нарушителя на СОЦ с целью расчета контролирующего кода горизонтальной целостности строки защищенной таблицы в рамках модели нарушителя сводится к атаке А2.

Доказательство. Нарушитель, действуя в рамках модели, может исследовать программные компоненты АИС уровня БД. Следовательно, ему известно, что целостность строк проверяется с помощью пакета, входящего в состав СОЦ. Изменение целостности данного пакета, является атакой А2.

Нарушитель может исследовать данный пакет, следовательно ему известно, что целостность строк контролируется с помощью значения хеш-функции $h( )$ от

конкатенации значений защищаемых столбцов таблицы *TAB* с добавлением значения ККИ.

Далее нарушитель для реализации атаки должен:

- либо изменить код пакета, в котором реализована *h( )* в серверной части АИС уровня СУБД, что является атакой *A2*;
- либо рассчитать правильный код контроля целостности, узнав значение ККИ по его названию, полученному при исследовании пакета.

Нарушитель может исследовать пакет, входящий в состав СОЦ, в который передается название ККИ для получения его значения. Следовательно, ему известно, что значения контейнеров хранятся в зашифрованном виде в таблице на неком значении, которое пакет в процессе функционирования получает от монитора безопасности. Шифрование производится с помощью функций *enc( )* и *dec( )*. Изменение нарушителем кода пакета (пакетов), в котором реализованы *enc( )* и *dec( )* в серверной части АИС уровня СУБД является атакой *A2*.

Нарушитель может исследовать программный пакет, входящий в состав СОЦ, в котором содержится код монитора безопасности. Следовательно, ему известно, что закрытое значение передается по протоколу Диффи-Хеллмана от библиотеки уровня ОС *cps_sm.so*, разработанной на языке *C/C++*.

Согласно модели, нарушитель не может исследовать под отладчиком состояние серверной системы и узнать значение мастер-ключа.

Доказательство завершено.

Утверждение 8. Атака нарушителя на СОЦ с целью расчета контролирующего кода целостности критичных программных компонентов БД в рамках модели нарушителя сводится к атаке A2.

Доказательство. Нарушитель, действуя в рамках модели, может исследовать программные компоненты АИС уровня БД. Следовательно, ему известно, что целостность критичных программных компонентов рассчитывается и проверяется с помощью пакета, входящего в состав СОЦ, в котором содержится код монитора безопасности. Изменение кода этого пакета является атакой *A2*.

Нарушитель может исследовать данный пакет, следовательно ему известно, что целостность критичных программных компонентов контролируется с помощью значения хеш-функции *h( )* от конкатенации строк пакета с добавлением значения, которое рассчитывается на основе значения, передающегося пакету по протоколу Диффи-Хеллмана от библиотеки уровня ОС *cps_sm.so*, разработанной на языке *C/C++*.

Изменение кода пакета, в котором реализована *h( )* в серверной части АИС уровня СУБД, является атакой *A2*.

Согласно модели нарушитель не может исследовать под отладчиком состояние серверной системы и узнать значение мастер-ключа.

Доказательство завершено.

Утверждение 9. Любая атака, в рамках модели нарушителя на защищенную информацию серверной части БД АИС, является атакой *A1* или *A2* либо сводится к атаке *A2*.

Доказательство следует из утверждений 7, 8 и информационного правила Кодда, которое гласит, что вся информация, хранимая в реляционной базе данных, должна быть явно, на логическом уровне, представлена единственным образом: в виде значений в реляционных таблицах.

Из утверждений 6 и 9 следует, что предлагаемое взаимодействие компонентов СОЦ делает систему с внедренной СОЦ устойчивой к атакам со стороны нарушителя, действующего согласно принятой модели.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Конявский В.А.* Методы и аппаратные средства защиты информационных технологий электронного документооборота: Дис… докт. техн. наук. – М., 2005.

2. *Архипочкин Е.В.* Построение системы обеспечения целостности информации, обрабатываемой в автоматизированной информационной системе, имеющей в своем составе СУБД. Информатизация и глобализация социально-экономических процессов: Сборник научных трудов II Международной научно-практической конференции (21 ноября 2007 года, Москва, Россия). – М.: ВНИИПВТИ, 2007. – С. 219-221.

УДК 004.414.2

В.С. Верба, В.А. Михеев

СИСТЕМНЫЙ АНАЛИЗ МЕТОДОВ ПРОЕКТИРОВАНИЯ МНОГОФУНКЦИОНАЛЬНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В настоящее время наличие в территориально-распределенной холдинговой структуре многофункциональной информационной системы, позволяющей построить единое информационное пространство, является не только требованием сегодняшнего дня, но и явным конкурентным преимуществом. Создание и внедрение многофункциональной информационной системы служит эффективным инструментом для выработки и проведения в жизнь согласованной общекорпоративной стратегии и тактики управления всеми предприятиями, входящими в состав холдинга.

Сложность и актуальность проблемы создания и внедрения крупных территориально-распределенных многофункциональных информационных систем, требует выработки рациональных проектных решений, включающих целый комплекс проблем.

При проектировании многофункциональной информационной системы следует учитывать следующие основные особенности, характерные для таких систем:

- сложность структуры – система состоит из множества подсистем, имеющих многофункциональные разветвленные взаимосвязи друг с другом и внешней средой;
- гетерогенность – система состоит из множества различных информационно-вычислительных, телекоммуникационных, программных и прочих ресурсов;
- рассредоточенность – система состоит из территориально-распределенных подсистем, которые располагаются на расстоянии нескольких тысяч километров друг от друга;
- динамика – система находится в стадии постоянного развития и совершенствования, подвергаясь при этом воздействию со стороны внешних и внутренних факторов и, вследствие этого, – постоянным модернизациям;
- многофункциональность – система предназначена для достижения большого количества целей, часто противоречащих друг другу;
- защищенность – в системе циркулирует как общедоступная информация, так и информация ограниченного доступа, что накладывает ряд дополнительных требований и ограничений.

Учитывая вышеизложенное, процесс проектирования многофункциональной информационной системы, должен быть организован таким образом, чтобы проектируемая информационная система [1]: