УДК 004.942

С. Ю. Бакулин

Московский инженерно-физический институт
Каширское шоссе, 31, Москва, 115409, Россия
E-mail: serguey.bakulin@gmail.com

АНАЛИЗ СОВМЕСТИМОСТИ ИНТЕГРИРУЕМЫХ ИНФОРМАЦИОННЫХ СИСТЕМ ПРИ ПОМОЩИ Σ-ИСЧИСЛЕНИЯ

В статье рассматривается проблема интеграции корпоративных приложений и ставится задача анализа интегрируемых приложений при помощи построения формальной модели предприятия с использованием ς-исчисления.
Ключевые слова: сигма-исчисление, теория объектов, интеграция приложений.

Одна из задач, которая обязательно встанет в ходе интеграционного проекта, – изучение комплекса существующего на предприятии программного обеспечения с целью обоснования возможности налаживания электронного взаимодействия всех этих систем и включения их в единую среду исполнения бизнес-процессов. Для этого предлагается построить модель предприятия с точки зрения используемых на предприятии информационных систем. Эта модель должна описывать предоставляемые каждой интегрируемой системой данные и их структуру, кроме того, модель должна явно указывать, каким способом можно извлечь необходимые данные из информационных систем. Пусть задано, что используется интеграционный брокер, обращающийся к внешним системам при помощи адаптеров. Тогда, входящий уровень абстракции брокера позволяет смотреть на данные, получаемые из внешних систем, как на объекты некоторых классов [1]. Классы, как известно, определяют структуру и поведение своих экземпляров. Не затрагивая сейчас темы поведения объектов, отметим, что структура классов будет соответствовать той структуре, в которой данные нам предоставляет (или из которого мы их извлекаем) интегрируемая система. Таким образом, внешняя, интегрируемая система для нас становится «черным ящиком», предоставляющим свои *точки доступа*. А это в точности способ, которым смотрит на компоненты теория объектов, которую и предлагается использовать в качестве инструмента построения модели комплекса систем предприятия.

Адаптируем определения теории объектов к рассматриваемому случаю интеграции информационных систем. *Точка доступа к сервисам* – набор методов, структура которых известна и которые можно выполнять извне. Точки доступа не включают в себя описание непосредственно исполняемого кода и заключенной в этот код логики. Точки доступа абстрактны, так как они определяют только прототипы для каждого метода, т. е. названия методов и типы аргументов и возвращаемых значений методов, а также общие описания того, как применять каждый метод [2]. Точки доступа бывают предоставляемые и требуемые. *Декларативное описание* – это описание точек доступа компонентов как набора абстрактных методов с указанием типов входных параметров и возвращаемых значений. Также в декларативное описание может входить краткое описание методов. Декларативное описание компонентов служит для построения общей компонентной архитектуры комплекса, описания ответственности каждой из компонент. При этом не рассматривается реализация компоненты и методов, входящих в предоставляемые ей точки доступа.

Операционное описание – это описание точек доступа компоненты, содержащее помимо заголовков методов еще и краткое описание алгоритма для каждого из этих методов. Операционное описание необходимо для более детального проектирования компонент системы, их функциональности [2]. Все системы описываются с точки зрения представляемых точек доступа к сервисам. Для описания точек доступа используется любой формальный подход. В мировой практике для этого начинает применяться ς-исчисление. Для простоты будем использовать ς-исчисление с типами. Это возможно, так как точки доступа компонента содержат методы, как объекты классической теории объектов. Конечно, для полного соответствия

объектам необходимо, чтобы точки доступа содержали еще и свойства и, в принципе, простые точки доступа можно было бы представлять как набор свойств, но для общности мы этого делать не будем. С другой стороны, наследование и полиморфизм, так как другие элементы этой теории в нашем случае не используются, что не ограничивает общность.

Будем считать, что

Интерфейс = {*Метод*$_1$, *Метод*$_2$,..., *Метод*$_N$}

Метод$_1$ = ς(x1) λ(*Параметр11*:*Tun11*), λ(*Параметр12*:*Tun12*),..., λ(*Параметр1m*:*Tun1m*) *Возврат*: *Tun*

Тип, *m* определяются для каждого метода. Аналогично описывается требуемый интерфейс *ИнтерфейсТреб*. Далее *Интегрируемый Компонент* =

{*Интерфейс*$_1$,..., *Интерфейс*$_k$, *ИнтерфейсТреб*$_1$,... *ИнтерфейсТреб*$_j$}.

Затем мы можем сформулировать операции над точками доступа.

- Операция 1. Равенство точек доступа.

Пусть есть две точки доступа:

Интерфейс$_1$ = {*Метод*$_{11}$, ..., *Метод*$_{2N}$} и *Интерфейс*$_2$ = {*Метод*$_{21}$, ..., *Метод*$_{2M}$}.

Интерфейс$_1$ = *Интерфес*$_2$, *если выполняется следующее условие*:

Интерфейс$_1$+ *Интрфейс*$_2$ = *Интерфейс*$_1$ = *Интерфейс*$_2$.

Необходимо определить сложение точек доступа и равенство методов.

- Операция 2. Равенство методов.

Будем говорить, что метод

Метод$_1$ = ς(x$_1$) λ(*Параметр*$_{11}$:*Tun*$_{11}$)...λ(*Параметр*$_{1m1}$:*Tun*$_{1m1}$) $b_1 : rt_1$ равен методу *Метод*$_2$ = ς(x$_2$) λ(*Парамерт*$_{21}$: *Tun*$_{21}$)...λ(*Параметр*$_{2m2}$: *Tun*$_{2m2}$) $b_2 : rt_2$, если

Параметр$_{1i}$ = *Параметр*$_{2i}$, $rt_1 = rt_2$, $m_1 = m_2$, *Tun*$_{1i}$ = *Tun*$_{2i}$ для любых $i = 1, \ldots m_1$.

Определим операцию сложения точек доступа.

- Операция 3. Сложение точек доступа.

Пусть есть две точки доступа:

Интерфейс$_1$ = {*Метод*$_{11}$, ..., *Метод*$_{1N}$} и *Интерфейс*$_2$ = {*Метод*$_{21}$, ..., *Метод*$_{2M}$}.

Тогда суммой точек доступа

Интерфейс$_1$ + *Интерфейс*$_2$

называется

Интерфейс$_3$ = {*Метод*$_{31}$, ... *Метод*$_{3z}$},

Метод$_{3i}$ ∈ {*Метод*$_{11}$, ..., *Метод*$_{1N}$, *Метод*$_{21}$, ..., *Метод*$_{2M}$} и при этом

Метод$_{3x}$ ≠ *Метод*$_{3y}$ для любых *x* и *y*.

Далее определим неравенство методов.

- Операция 4. Неравенство методов.

Будем говорить, что метод

Метод$_1$ = ς(x$_1$) λ(*Параметр*$_{11}$:*Tun*$_{11}$)...λ(*Параметр*$_{1m1}$:*Tun*$_{1m1}$) b_1: rt_1 не равен методу *Метод*$_2$ = ς(x$_2$) λ(*Параметр*$_{21}$:*Tun*$_{21}$)...λ(*Параметр*$_{2m2}$:*Tun*$_{2m2}$) b_2: rt_2,

если хотя бы одно из условий

Параметр$_1$ = *Параметр2*$_2$, $rt_1 = rt_2$, $m_1 = m_2$, *Tun*$_{1i}$ = *Tun*$_{2i}$ ($i = 1 \ldots m_1$)

не выполняется.

- Операция 5. Вхождение интерфейсов.

Будем говорить, что интерфейс *ИнтерфейсТреб* входит в интерфейс *Интерфейс1*, если для любого метода из *ИнтерфейсТреб* найдется метод *Интерфейс* такой, что эти методы тождественно равны. Пусть

ИнтерфейсТреб = {*МетодТреб*$_1$, ... *МетодТреб*$_N$},

Интерфейс = {*Метод*$_1$, ... *Метод*$_M$}.

Тогда *ИнтерфейсТреб* входит в *Интерфейс*, если

$\forall i \in [1, n] \;\; \exists j \in [1, m] :$ *МетодТреб*$_i$ = *Метод*$_j$.

Вхождение требуемого интерфейса в предоставляемый определяет возможность взаимодействия компонентов, т. е. если компонент M_1 имеет требуемый интерфейс *ИнтерфейсТреб*, а компонента M_2 – предоставляемый интерфейс *Интерфейс*, и при этом *ИнтерфейсТреб* входит в *Интерфейс*, то можно сказать, что компоненты M_1 и M_2 взаимодействуют корректно [2]. Этот подход в первую очередь позволит абстрагироваться от типа и структуры интер-

фейса. Это может быть web-метод, API или просто файл на жестком диске. В терминах предложенного аппарата это просто метод со своими аргументами и типом возвращаемого значения. Кроме того, такой подход позволит на этапе анализа существующих систем выявить отсутствие необходимых компонентов.

Покажем пример построения модели для рассматриваемого комплекса систем.

Если мы имеем компонентную модель, то, используя формальные методы, можем определить, насколько корректно могут взаимодействовать системы комплекса, т. е. если все требуемые точки доступа входят в соответствующие предоставляемые точки, то не возникнут коллизии при передаче и обработке данных в рассматриваемой информационной системе. Опишем компонет ISCPerson, представляющий собой базу данных сотрудников компании. Для простоты изложения здесь и далее уберем незначимые интерфейсы реального компонента, связанные с отпусками, сервисными функциями и др.

ISCPerson =

[СоздатьСотрудника = ς(s) λ(Имя : *String) λ(Фамилия* : *String) λ(Пароль* : *String) λ(Отделение* : *{“SE”,”Sales”,”Support”,”CO”,”TM”,”M”}) λ(Должность* : *String) λ(Телефон* : *String) λ(Почта* : *String) λ(Подразделение* : *String) λ(Отпуск* : *Integer) λ(Удален* : *Boolean)* b_{cc} *: Integer*;

СоздатьОтдел = ς(s) λ(Название : *String) λ(Менеджер* : *Integer) λ(Подчинение* : *Integer)* b_{co} *: Integer*;

ИзменитьПароль = ς(s) (Идентификатор : *Integer) λ(СтарыйПароль* : *String) λ(НовыйПароль* : *Integer)* $b_{измп}$ *: Integer*;

СотрудникПоИдентификатору = ς(s) (Идентификатор : *Integer)* $b_{спи}$ *: ResultSet*;

СотрудникПоИмени = ς(s) (Имя^ String) $b_{спим}$ *^ ResultSet*; *]*

Очевидно, что этот компонент просто предоставляет различные методы для работы с базой данных сотрудников. Рассмотрим любой компонент, который использует базу данных сотрудников компании. Для нашего примера очень характерна ситуация, когда появился новый сотрудник. Этот сотрудник вносится в разные системы различным образом, поскольку в каждой из них он имеет свою природу и структуру. Рассмотрим, например, точки доступа, предоставляемые CRM системой.

TRS.Crm =

[СоздатьПользователя = ς(s) λ(Имя : *String) λ(Фамилия* : *String) λ(Логин* : *String) λ(Пароль* : *String) λ(Отделение* : *{“1”,”2”,”3”,”4”,”5”,”6”,”7”}) λ(Должность* : *String) λ(Удален* : *Boolean)* $b_{сп}$ *: Integer*;

СоздатьОтдел = ς(s) λ(Название : *String)* $b_{сп}$ *: Integer*;

ИзменитьПользователя = ς(s) λ(Идентификатор : *Integer)λ(Имя* : *String) λ(Фамилия* : *String) λ(Логин* : *String) λ(Пароль* : *String) λ(Отделение* : *{“1”,”2”,”3”,”4”,”5”,”6”,”7”}) λ(Должность* : *String) λ(Удален* : *Boolean)* $b_{ип}$ *: Integer*;

ИзменитьОтдел = ς(s) λ(Идентификатор : *Integer) λ(Название* : *String)* b_{uo} *: Integer*;

Запрос = ς(s) λ(Текст : *String) λ(ИмяПользователя* : *String) λ(Пароль* : *String)* $b_з$ *: ResultSet*; *]*

В этом компоненте также есть такая сущность, как пользователь, но, естественно, структура его отличается. Кроме имени и фамилии, компонент требует имени пользователя и пароля, поскольку предполагается удаленный доступ к этому компоненту из незащищенных сетей. Кроме того, в структуре, принятой в этой системе, отделы нумеруются, а не называются по их буквенным кодам. Наконец, принято дополнительное обозначение для еще не определившихся сотрудников – вынесено в отдельный номер отделение по умолчанию. Наконец, ряд параметров в аналогичном методе просто отсутствует. Теперь построим операционное описание этого же компонента.

TRS.Crm =

[СоздатьПользователя = ς(s) λ(Имя : *String) λ(Фамилия* : *String) λ(Логин* : *String) λ(Пароль* : *String) λ(Отделение* : *{“1”,”2”,”3”,”4”,”5”,”6”,”7”}) λ(Должность* : *String)*

λ(Удален : Boolean) (TRS.CrmТреб.СоздатьСотрудника(Имя, Фамилия, Почта, Пароль, FN ConvertDepartment(Отделение), Должность, Удален)) : Integer

СоздатьОтдел = ς(s) λ(Название : String)(TRS.CrmТреб.СоздатьОтдел(Название)) : Integer

ИзменитьПользователя = ς(s) λ(Идентификатор : Integer)λ(Имя : String) λ(Фамилия : String) λ(Логин : String) λ(Пароль : String) λ(Отделение : {"1","2","3","4","5","6","7"}) λ(Должность : String) λ(Удален : Boolean) (TRS.CrmТреб.ИзменитьСотрудника(Имя, Фамилия, Почта, Пароль, FN ConvertDepartment(Отделение), Должность, Удален)) : Integer

СоздатьДействие = ς(s) λ(Название : String) λ(ДатаСоздания : Date) λ(ДатаЗакрытия : Date) λ(Описание : String) λ(Тип : String) λ(Оповещение : String) λ(Удален : Boolean) $b_{сд}$: Integer;

ИзменитьОтдел = ς(s) λ(Идентификатор : Integer) λ(Название : String) (TRS.CrmТреб.ИзменитьОтдел(Идентификатор, Название)) : Integer

Запрос = ς(s) λ(Текст : String) λ(ИмяПользователя : String) λ(Пароль : String) $b_з$: ResultSet; *]*

Применяя аналогичные приемы для всех методов интерфейса, мы обнаруживаем, что вполне можем определить требуемый интерфейс для нашего компонента. Итак, исходя из операционного описания можно построить формальное описание интерфейса *TRS.CrmТреб*, который определяет необходимые для целостной работы комплекса систем точки доступа.

TRS.CrmТреб =

[СоздатьПользователя = ς(s) λ(Имя : String) λ(Фамилия : String) λ(Логин : String) λ(Пароль : String) λ(Отделение : {"1","2","3","4","5","6","7"}) λ(Должность : String) λ(Удален : Boolean) $b_{сп}$: Integer;

СоздатьОтдел = ς(s) λ(Название : String) $b_{со}$: Integer;

ИзменитьПользователя = ς(s) λ(Идентификатор : Integer)λ(Имя : String) λ(Фамилия : String) λ(Логин : String) λ(Пароль : String) λ(Отделение : {"1","2","3","4","5","6","7"}) λ(Должность : String) λ(Удален : Boolean) $b_{ип}$: Integer;

ИзменитьОтдел = ς(s) λ(Идентификатор : Integer) λ(Название : String) $b_{ио}$: Integer;

Запрос = ς(s) λ(Текст : String) λ(ИмяПользователя : String) λ(Пароль : String) $b_з$: ResultSet;

СоздатьДействие = ς(s) λ(Название : String) λ(ДатаСоздания : Date) λ(ДатаЗакрытия : Date) λ(Описание : String) λ(Тип : String) λ(Оповещение : String) λ(Удален : Boolean) $b_{сд}$: Integer; *]*

Видно, что двух методов нам еще не закрыли предоставляемыми интерфейсами, в то время как оставшиеся методы полностью входят в предоставляемые точки доступа. Обратимся к другим компонентам за необходимым «строительным материалом».

Аналогичный анализ других компонентов предприятия показывает, в частности, что компонент WRC в достаточной степени наделен необходимыми точками доступа. Рассмотрим их поподробнее.

WRC =

[СоздатьПроблему = ς(s) λ(Название : String) λ(ДатаСоздания : Date) λ(ДатаЗакрытия : Date) λ(Описание : String) λ(Версия : String) λ(Клиент : Integer) λ(Контакт : Integer) λ(СледующееДействие : Date) $b_{создп}$: Boolean;

СоздатьДействиеВПроблеме = ς(s) λ(Название : String) λ(ДатаСоздания : Date) λ(Описание : String) λ(Тип : String) $b_{сдвп}$: Integer;

ОсуществитьПоиск = ς(s) λ(Запрос : String) $b_{осп}$: ResultSet;

...{остальные элементы интерфейса}]

Тогда интерфейс TRS.Crm модифицируется следующим образом.

TRS.Crm =

[СоздатьПользователя = ς(s) λ(Имя : String) λ(Фамилия : String) λ(Логин : String) λ(Пароль : String) λ(Отделение : {"1","2","3","4","5","6","7"}) λ(Должность : String) λ(Удален : Boolean) (TRS.CrmТреб.СоздатьСотрудника(Имя, Фамилия, Почта, Пароль, FN ConvertDepartment(Отделение), Должность, Удален)) : Integer

СоздатьОтдел = ς(s)
λ(Название : *String)(TRS.CrmТреб.СоздатьОтдел(Название))* : *Integer*
ИзменитьПользователя = ς(s) λ(Идентификатор : *Integer)λ(Имя* : *String)*
λ(Фамилия : *String) λ(Логин* : *String) λ(Пароль* : *String)*
λ(Отделение : *{"1","2","3","4","5","6","7"}) λ(Должность : String) λ(Удален : Boolean)*
(TRS.CrmТреб.ИзменитьСотрудника(Имя, *Фамилия*, *Почта*, *Пароль*, *FN*
ConvertDepartment(Отделение), *Должность*, *Удален)) : Integer*
СоздатьДействие = ς(s) λ(Название : *String) λ(ДатаСоздания* : *Date)*
λ(ДатаЗакрытия : *Date) λ(Описание* : *String) λ(Тип* : *String) λ(Оповещение* : *String)*
λ(Удален : *Boolean) (TRS.CrmТреб.СоздатьДействиеВПроблеме(Название*, *ДатаСоздания*,
ДатаСоздания, *Описание*, *FN ConvertTypes(Тип)*, *0*, *0)) : Integer*;
ИзменитьОтдел = ς(s) λ(Идентификатор : *Integer) λ(Название* : *String)*
(TRS.CrmТреб.ИзменитьОтдел(Идентификатор, *Название))* : *Integer*
Запрос = ς(s) λ(Текст : *String) λ(ИмяПользователя* : *String) λ(Пароль* : *String)*
(TRS.CrmТреб.ОсуществитьПоиск(Запрос,"_system","sys")) : *ResultSet*; *]*

Приводить модифицированную требуемую точку доступа уже не нужно, достаточно обратить внимание на то, что требуемые методы найдены и также входят в предоставляемые компонентами методы. Таким образом, суммы точек доступа *ISCPerson + WRC* достаточно, чтобы образовать суммарный набор методов, в который *входит* требуемая точка доступа TRS.Crm, поэтому можно считать доказанным, что взаимодействие систем с точки зрения синхронизации пользователей в комплексе возможно и обеспечивается существующими точками доступа.

Заключение

Используя указанную методику, можно на этапе анализа предприятия проверить совместимость используемых информационных систем и их пригодность для осуществления интеграции при помощи современных средств, таких как интеграционные брокеры. Так как интеграционные проекты очень сложны и требуют значительного вложения ресурсов, подобный анализ позволит существенно сократить риски неуспешного внедрения интеграционного брокера путем своевременного выявления отсутствия необходимых интерфейсов или заложить в проект время на их разработку.

Список литературы

1. *Abadi M.* A theory of objects. N. Y.: Springer, 1996. 396 p.
2. *Seacord R., Plakosh D., Lewis G.* Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business. N. Y.: Addison-Wesley, 2003. 352 p.
3. *Stokes N.* EAI and beyond: A multi-level flow model // Business Integration. 2004. № 9. P. 66–70.

Материал поступил в редколлегию 06.11.2008

S. U. Bakulin

COMPATIBILITY OF INFORMATION SYSTEM COMPONENTS ANALYSIS USING Σ-CALCULATIONS

The article is devoted to the problem of enterprise application integration, the author states the task of integrated applications analysis using formal enterprise model on the base of ς-calculations.

Keywords: sigma-calculations, objects theory, enterprise applications integration.