Библиотека Инженера

«СОЛОН»

Фельдман Я. А.

# Создаем информационные системы

Разработка прикладных информационно-управляющих систем для предприятий, организаций и средней школы – это просто!


- Уникальный подход к разработке систем
- Пошаговая программная реализация модели данных
- Авторские примеры на компакт-диске

Находка для специалиста!

**УДК 621**
**ББК 30**
**Ф 39**

**Я. А. Фельдман**

**Создаем информационные системы.** — М.: СОЛОН-ПРЕСС, 2010. — 120 с.: ил. — (Серия «Библиотека инженера»)

**ISBN 5-98003-256-8**

В книге подробно описана новая модель данных и ее программная реализация, предложенная и выполненная автором. Цель — дать возможность каждому предприятию построить единую информационную систему высокого качества. Работающая программа и общий подход, описанный в книге, дают возможность решить эту задачу. Особе место в книге занимает применение системы в средней школе — как для тестирования учащихся, так и для управления всем педагогическим процессом.

***К книге прилагается компакт-диск с копиями экранов и демо-версией программы.***



По вопросам приобретения обращаться: **ООО «АЛЬЯНС-КНИГА КТК»**
Тел: (495) 258-91-94, 258-91-95, сайт **www.abook.ru**
Сайт издательства «СОЛОН-ПРЕСС»: **www.solon-press.ru**
E-mail: **solon-avtor@coba.ru**

**ISBN 5-98003-256-8**

# Заказчик

## 6. Вначале были деньги

*Ищу я выход из ворот, но нет его —*
*есть только вход, и то не тот.*
*В. С. Высоцкий*

Представьте себе, что сегодня утром вы стали хозяином предприятия.

*Предприятием я называю группу людей, долговременно и согласованно стремящихся к общему результату. Размер группы, ее юридический статус и профиль работы (школа, завод, поликлиника, торговая сеть) не имеют никакого значения.*

Представьте себе, что у вас на предприятии абсолютная власть и что денег у вас немеряно. При этом вы уважаете компьютеры и все, что с этим связано, и очень хотите их применить. Что будете делать?

Во-первых, вы купите компьютеры (10? 20? 150? 380? 1500?), после чего ваши сотрудники зависнут в играх и пасьянсах.

Во-вторых, вы установите сети и оплатите Интернет, после чего одни ваши сотрудники зависнут в почте, другие в чатах и пр. и др.

А дальше? Дальше ситуация будет развиваться по одному из следующих сценариев.

***Сценарий первый.*** Пускаем дело *на самотек*. Каждый из сотрудников начинает сам добывать программы (пиратские копии), сам ставить, сам изучать, сам заводить данные. *Не позднее чем через полгода* взору удивленного наблюдателя открывается живописная картина под условным названием «***зоопарк программ и данных***». Надо ли описывать муки какого-нибудь начальника отдела, знающего, что *эти данные точно есть*, но не в силах отыскать, *где именно они есть*, и, наконец, *вводящего* их в систему *еще раз*? Надо ли описывать бесконечные распечатки на бумагу с одного компьютера, сделанные только для того, чтобы кто-то другой набирал эти цифры заново, чтобы ввести их в другой компьютер, стоящий рядом? *Душераздирающее зрелище*, как сказал бы Иа-Иа.

***Сценарий второй.*** Набираем студентов. Для ровного счета человек семь. Назначаем начальника из родственников. Начинаем лепить самодельный софт. Если учесть, что студентов обучали по учебным планам, утвержденным в Москве одиннадцать лет назад, и что учились они не очень, и что все ошибки у них еще впереди, что получим? *Оно, конечно, работает, но пользоваться им невозможно.* А никто и не пользуется. И не надо было включать.

***Сценарий третий.*** Приглашаем серьезных профессионалов. Microsoft, например, или Oracle, или SAP. Платим. Они ставят. Оно работает. Но не удобно. Чтобы было удобно, надо платить еще. И еще. И еще. В конце концов, кончаются деньги. И дальше — смотри выше.

Есть еще ***четвертый сценарий***, когда покупаются отдельные подсистемы: 1С для бухгалтерии, ПАРУС для производства, ФОКУС-ПОКУС для управления складом и др. (названия и подсистемы выбраны условно). Но системы из коробки просто так не работают — их надо привязывать и настраивать. При этом получается гремучая смесь из предыдущих трех вариантов.

Что же делать? Где выход? Выходом является технология FTS.

Рассмотрим ***минимальный вариант***. У вас есть ***один*** *надежный* человек, способный после некоторой подготовки стать администратором вашей FTS-системы. Предположим, что он не имеет никаких знаний по программированию, но способен переписывать слова латиницей, не теряя при этом букв. Если он хочет и может учиться, мы *за месяц* (двадцать рабочих дней) сделаем из него *нормального* администратора FTS. Упомянутые 5% ситуаций, для которых нужны программисты, может взять на себя служба поддержки.

Вы не обязаны допускать службу поддержки к реальным данным вашей фирмы. Достаточно держать «демобазу», на которой администратор будет демонстрировать службе поддержки свои пожелания и проблемы и на которой он будет проверять новые версии, чтобы самостоятельно перенести их на основную базу. Для этого ему достаточно открыть демобазу в Интернет, и служба поддержки будет работать с ней удаленно. При этом основная база остается закрытой для внешнего мира.

***Максимальный вариант.*** Предположим дополнительно, что у вас есть два трудолюбивых студента, один очень умный, другой не очень. Очень умного обучаем Java, не очень умного SQL. После чего служба поддержки перестает работать над вашей базой (исчезают те самые 5%) и только отвечает студентам на их вопросы.

Между минимальным вариантом (один НЕ программист) и максимальным вариантом (один НЕ программист и два программиста) есть *промежуточные* (один НЕ программист и один программист; один программист, выполняющий работу администратора и пр.)

*Но в любом случае вам понадобится не более трех человек* с не очень большой зарплатой.

***Внимание!*** *Надежность администратора должна быть абсолютной. Но это вопрос чести, а честь не купишь ни за какие деньги.*

Нарисованная здесь картина настолько привлекательнее текущих реалий, что напрашивается очевидный вопрос: ***за счет чего возможен такой фантастический выигрыш в эффективности***?

Ведь даже студенты из второго варианта это не два, а семь человек! Что же говорить об остальных вариантах?

Ответ уже дан выше. *Гибкость* системы позволяет *наращивать ее понемногу*, изо дня в день. В то время как *обычно* подсистемы добавляют большими блоками один раз в *три-четыре-шесть* месяцев. Большая толпа программистов нужна для того только, чтобы в пиковую неделю приемки очередной подсистемы успеть наложить все заплатки на все дыры, обнаруженные заказчиком. О качестве такой «подсистемы в заплатках» можно только догадываться. Все же остальное время программистам нечего делать, кроме как висеть в чате или слоняться из угла в угол. Но если вы вздумаете их уволить, то к очередной пиковой неделе вы не соберете нужной толпы.

Вот такая арифметика. Для пущей убедительности приведу диаграмму, на которой показан перечень работ и относительные трудозатраты для некоторой абстрактной подсистемы, по материалам фирмы IBM.

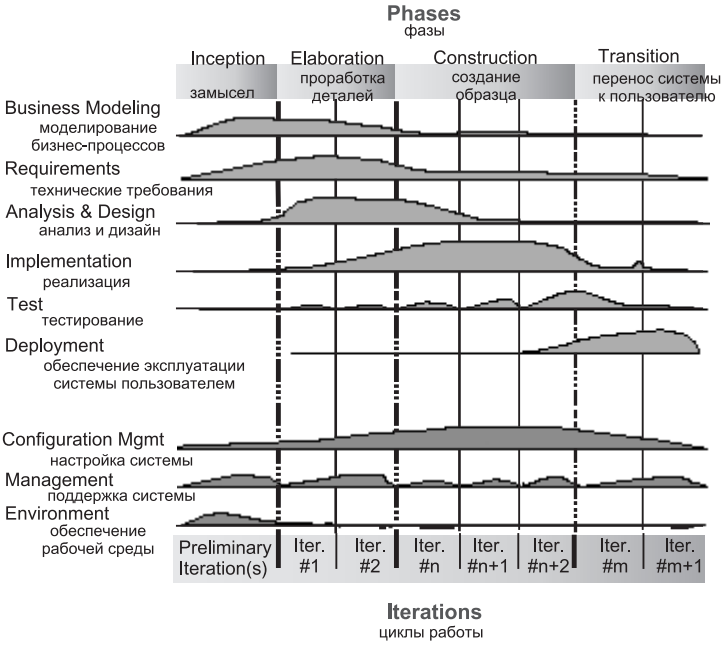


*Рис. 2. Трудоемкость во времени*

Время, показанное на диаграмме по горизонтали, по мнению IBM, для одной подсистемы составляет от шести до восемнадцати месяцев. У нас единичное добавление занимает от пяти минут до, самое большее (в случае Java-функций, разработанных заново), трех дней.

# Архитектор

## 7. Объектно-ориентированный подход и его конкуренты

Жизнь информационных систем протекает в четырех *временных горизонтах* (смотри следующую таблицу).

*Таблица 2*

*Временные горизонты*

| Название | Временной горизонт | Что нового? |
|---|---|---|
| Техника | 1—2 года | Программа, процедура |
| Технология | 3—5 лет | Язык, СУБД |
| Методология | 7—10 лет | Архитектура |
| Идеология | 15—50 лет | Общий подход |

Сейчас нас интересует **общий подход**. Таких подходов на сегодня используется несколько, и объектно-ориентированный подход (ООП) — только один из них. Наиболее известные подходы перечислены в следующей таблице.

*Таблица 3*

*Общие подходы*

| Подход (стандарт) | Единица работы |
|---|---|
| ООП | Объект |
| IDEF3 | Поток работ |
| DFD | Поток данных |
| IDEF0 | Функция |

Чтобы определить, как соотносятся между собой эти подходы, рассмотрим следующий рисунок (рис. 3).

При создании информационных систем происходит с одной стороны (на рисунке — сверху) осмысление и структуризация

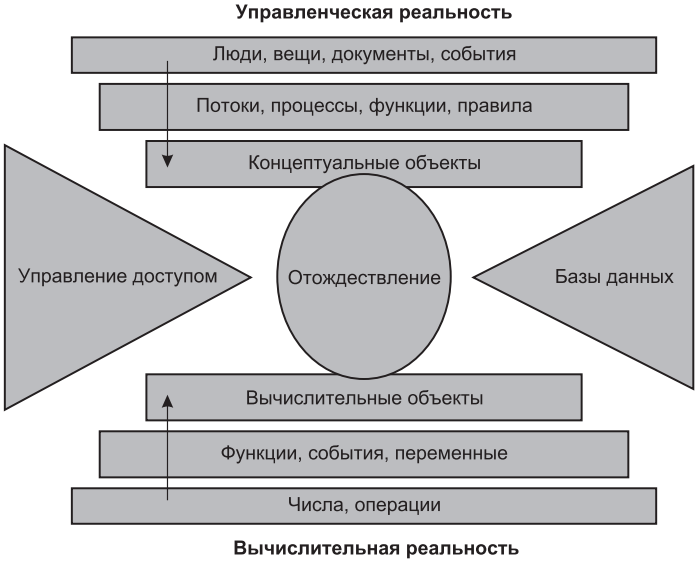


*Рис. 3. ООП сегодня*

управленческой реальности, с другой стороны (на рисунке — снизу) — обогащение вычислительной среды. Объектно-ориентированный подход доводит дело снизу — до *вычислительных объектов* (объектно-ориентированное программирование; такие языки, как Java и C++), а сверху до *концептуальных объектов* (объектно-ориентированный дизайн, например, с использованием Rational Rose UML). В конце концов, вычислительные объекты строятся как реализация объектов концептуальных, то есть происходит (как показано на рисунке) их фактическое отождествление.

Другие общие подходы просто не доводят структуризацию управленческой реальности до такой глубины. И хотя в том понимании ООП, которое господствует сегодня и которое показано на рис. 3, есть слабые места, замена его на другие подходы из таблицы 2 была бы шагом назад.

Обратите внимание на еще две компоненты, фактически не интегрированные в идеологию ООП, но неизбежно присутствующие в реальной работе. Это Управление доступом и База данных, показанные на рис. 3 двумя треугольниками.

Система FTS предлагает, во-первых, интегрировать эти две компоненты в общий подход и, во-вторых, вместо механического отождествления концептуальных и вычислительных объектов построить некий новый вид объектов (FTS-объекты), равно удаленный от тех и от других.

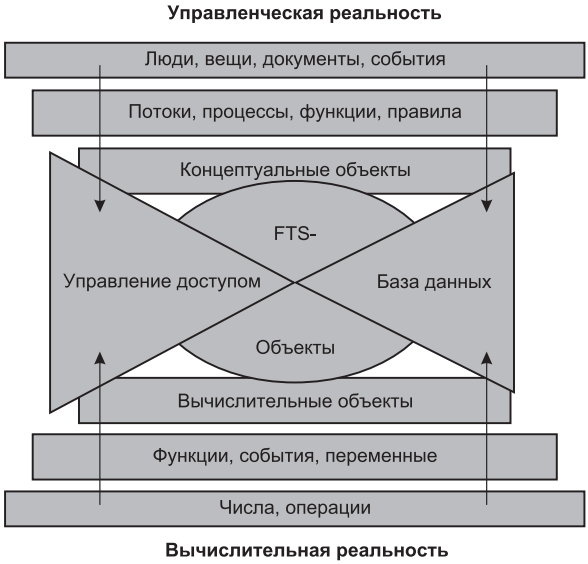


*Рис. 4. FTS-подход*

# Содержание