ISSN 1998-0663

№2(04)—2008

www.bi-network.ru/jbi/

БИЗНЕС-ИНФОРМАТИКА

МЕЖДИСЦИПЛИНАРНЫЙ НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

BUSINESS INFORMATICS

Учредитель:
Государственный университет—Высшая школа экономики

Главный редактор
Никитин В.В.

Заместители главного редактора
Горбунов А.Р.
Ульянов М.В.

Редакционная коллегия:
Абдульраб А. (Франция)
Авдошин С.М.
Алескеров Ф.Т.
Белов В.В.
Вирин Ф.Ю.
Грибов А.Ю.
Громов А.И.
Гюнтер Х. (Германия)
Каменнова М.С.
Калягин В.А.
Козырев О.Р.
Кузнецов С.О.
Мальцева С.В.
Миркин Б.Г. (Великобритания)
Моттль В.В.
Пальчунов Д.Е.
Силантьев А.Ю.
Таратухин В.В.

В ЭТОМ НОМЕРЕ:

Программная инженерия

Анализ и моделирование бизнес-процессов

Информационные системы и технологии в бизнесе

Дискуссии

БИЗНЕС-
ИНФОРМАТИКА

http://www.bi-network.ru/jbi/ №2(04)-2008

СОДЕРЖАНИЕ

Программная инженерия
С.М. Авдошин, А.А. Савельева.
Проблемы оценки криптозащищенности информационных систем 3

В.В. Липаев.
Тестирование крупных комплексов программ на соответствие требованиям 16

Анализ и моделирование бизнес-процессов
А.Р. Горбунов
Системная динамика: моделирование принятия стратегических и оперативных решений (метод функции капитального класса).....25

Информационные системы и технологии в бизнесе
Е.О. Потапова.
Разработка методологии применения средства Solution Manager в задаче интеграции системы SAP 35

О.Р. Набиуллин, В.М. Норкин.
Архитектура высокопроизводительной системы многоагентного моделирования 48

Дискуссии
М.М. Крюков.
Игровое моделирование кризисных явлений и структурных преобразований в экономике 61

БИЗНЕС-ИНФОРМАТИКА
№2(04)–2008

Междисциплинарный
научно-практический журнал

Подписной индекс издания
в каталоге агентства
«Роспечать» –72315

Главный редактор
Никитин В.В.

Учредитель:
Государственный университет-
Высшая школа экономики
© Государственный университет-
Высшая школа экономики

Выходит 4 раза в год.

Редакция:

Зам. главного редактора
Горбунов А.Р.
Зам. главного редактора
УльяновМ.В.

Научный редактор
Лычкина Н.Н.

Технический редактор
Осипов В.И.

Дизайн обложки
Борисова С.Н.

Компьютерная вёрстка
Волков А.А.
Бортникене Е.И.

Администратор веб-сайта
Проценко Д.С.
http://www.bi-network.ru/jbi/

Адрес редакции:
105679, г. Москва,
ул. Кирпичная, д. 33.
Тел.+7 (095) 771-32-38,
e-mail: bijournal@hse.ru

За точность приведённых сведений
и содержание данных,
не подлежащих открытой
публикации,
несут ответственность авторы

При перепечатке
ссылка на журнал
«Бизнес-информатика» обязательна

Журнал отпечатан в типографии
ООО «ГЕО-ТЭК
г. Красноармейск Московской обл.
Тел: (495) 993-1623
Тираж 500 экз.

ПРОБЛЕМЫ ОЦЕНКИ КРИПТОЗАЩИЩЁННОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ

С.М. Авдошин,
профессор, руководитель отделения программной инженерии, заведующий кафедрой управления разработкой программного обеспечения Государственного университета – Высшей школы экономики, академик Всемирной академии наук комплексной безопасности, член-корр. Международной академии информатизации, Россия
savdoshin@hse.ru
А.А. Савельева,
преподаватель кафедры управления разработкой программного обеспечения отделения программной инженерии факультекта бизнес-информатики Государственного унивеститета-высшей школы экономики.
asavelieva@hse.ru

Рассмотрено построение классификаций криптографических средств, криптоаналитических атак и злоумышленников, действия которых направлены на взлом криптосистем. Построена модель угроз безопасности в конкретной компьютерной системе, как композиция модели злоумышленника, модели атак и модели криптосистемы.

Введение

Разработка новых подходов для анализа защищённости информационных систем – актуальная задача в области информационной безопасности. Особое место занимает проблема оценки криптографических средств защиты на предмет их соответствия потребностям организации.

Задача определения эффективности системы защиты информации (СЗИ) при использовании криптографических методов защиты зачастую более трудоёмкая, чем их разработка, требует наличия специальных знаний и более высокой квалификации [Ростовцев, 98]. Анализ криптостойкости шифра – научная, а не инженерная задача. Эффективность одной и той же криптосистемы в разных контекстах использования может существенно отличаться. Использование СЗИ, обеспечивающей устойчивость к взлому ниже некоторой «фоновой» вероятности экономически неоправданно [11]. Например, если вероятность выхода компании из бизнеса менее чем один из миллиона, нет смысла в защите информации, могущей нанести компании ущерб, сопоставимый с кризисом рынка, и использовать алгоритм, вероятность вскрытия которого за приемлемое время составляет 2^{-200}.

При разработке подходов к анализу криптографической защищённости информационной системы необходимо учитывать, каким угрозам подвергнется система со стороны злоумышленников. В первом разделе приведены результаты систематизации криптографических средств защиты и анализа их функциональных возможностей. Во втором разделе описана классификация существующих типов криптоатак с учётом различных критериев. Последний раздел посвящён построению модели злоумышленника, заданной с помощью ряда параметров. Разработанные классификации позволяют определить зависимость атак, которым может подвергаться криптосистема, от области её использования.

Классификация криптографических средств

Существуют несколько способов, в соответствии с которыми могут классифицироваться криптографические системы. В классификационную схему, предлагаемую в данной работе, включены критерии, важные с точки зрения оценки практической эффективности криптосистемы. Разработанная классификация позволяет определить подверженность криптосистемы тем или иным типам атак со стороны злоумышленника, идентифицируя её в соответствии с особенностями реализации. Криптографические средства различают по:

- ✦ доступности информации о криптоалгоритме:
  - ✧ криптосистемы ограниченного использования;
  - ✧ криптосистемы общего использования;
- ✦ количеству ключей:
  - ✧ бесключевые;
  - ✧ одноключевые;
  - ✧ двухключевые;
  - ✧ многоключевые;
- ✦ стойкости криптоалгоритма:
  - ✧ безусловно стойкие;
  - ✧ доказуемо стойкие;
  - ✧ предположительно стойкие;
- ✦ используемым средствам шифрования:
  - ✧ программные;
  - ✧ аппаратные;
  - ✧ программно-аппаратные;
- ✦ наличию сертификата:
  - ✧ сертифицированные;
  - ✧ несертифицированные.

Классификация по доступности информации о криптоалгоритме

В работе [24] описана следующая классификация криптосистем:

- ✦ криптосистемы ограниченного использования;
- ✦ криптосистемы общего использования.

Криптографическая система называется криптосистемой ограниченного использования, если её стойкость основывается на сохранении в секрете самого характера алгоритмов шифрования и дешифрования. Простейший пример такой системы – шифр Цезаря, в котором преобразование информации сводится к простой замене каждого символа открытого текста третьим следующим за ним символом алфавита. Секретность алгоритма не является большим препятствием, например, для определения типа программно реализованного криптографического алгоритма требуется лишь несколько дней инженерного анализа исполняемого кода.

Стойкость криптосистемы общего использования в соответствии со сформулированным Крикхоффсом фундаментальным допущением криптоанализа основывается не на секретности алгоритмов шифрования и дешифрования, а на секретности некоторого значения, называемою её ключом. Такой ключ должен вырабатываться конкретными пользователями таким образом, чтобы даже разработчик криптосистемы не мог раскрыть её, не имея доступа к ключу. Для некоторых применений (в военных, дипломатических и разведывательных ведомствах) у разработчика общей криптосистемы нет причин открытым образом описывать характер её алгоритмов. Сохраняя информацию об алгоритме в тайне, можно обеспечить дополнительную безопасность. Однако решающим обстоятельством, позволяющим полагаться на такую секретность, это не является, поскольку ничего нельзя сказать о том, когда она может быть скомпрометирована. По этой причине исследования надёжности таких систем всегда должны проводиться в предположении, что потенциальному противнику о криптосистеме известно всё, за исключением используемого ключа.

Как показывает Брюс Шнайер в своей статье [25], нежелание разработчиков открыто обсуждать криптоалгоритмы – это сигнал, что их продуктам нельзя доверять. Показателен приведённый в [26] пример программы eBook Pro. Она преподносилась разработчиками как «единственный программный продукт во вселенной, способный обеспечить Вашей информации практически 100% защиту от взлома». При анализе надёжности оказалось, что защита заключается в наложении при помощи операции XOR на каждый байт защищаемых данных последовательно всех байтов строки «encrypted», что эквивалентно наложению однобайтовой константы и обеспечивает возможность извлечения защищённой информации без подбора ключа или иных длительных операций.

Классификация по количеству ключей

По классификационной схеме, описанной в [14], криптосистемы подразделяются на:

- ✦ бесключевые, не использующие каких-либо ключей в процессе криптографических преобразований;
- ✦ одноключевые, использующие в своих вычислениях некий секретный ключ;
- ✦ двухключевые, в которых на различных этапах вычислений применяются два вида ключей: секретные и открытые.

В двухключевых криптосистемах для зашифрования и расшифрования используются различные функции. Асимметричные алгоритмы основаны на ряде математических проблем, на которых и базируется их стойкость. Пока не найден полиномиальный алгоритм решения этих проблем, данные алгоритмы будут стойкими.

Наиболее известные асимметричные криптосистемы:

- ✦ рюкзачная криптосистема (Knapsack Cryptosystem) [16];
- ✦ криптосистема RSA;
- ✦ криптосистема Эль-Гамаля – EGCS (El Gamal Cryptosystem);

- ✦ криптосистема, основанная на свойствах эллиптических кривых – ECCS (Elliptic Curve Cryptosystems).

Применение алгоритмов шифрования с открытым ключом позволяет:

- ✦ избавиться от необходимости секретных каналов связи для предварительного обмена ключами;
- ✦ свести проблему взлома шифра к решению трудной математической задачи, т.е. принципиально по-другому подойти к обоснованию стойкости криптосистемы;
- ✦ решать средствами криптографии задачи, отличные от шифрования, например, задачу обеспечения юридической значимости электронных документов.

Общепризнано подразделение одноключевых (симметричных) криптографических алгоритмов на следующие основные категории:

- ✦ блочные шифры;
- ✦ поточные шифры.

Идея, лежащая в основе большинства итерационных блочных шифров, состоит в построении криптографически стойкой системы последовательным применением относительно простых криптографических преобразований. Принцип многоразового шифрования с помощью простых криптографических преобразований впервые предложен Шенноном в работе [15]: он использовал с этой целью преобразования перестановки и подстановки. Первое из этих преобразований переставляет отдельные символы преобразуемого информационного блока, а второе заменяет каждый символ (или группу символов) из преобразуемого информационного блока другим символом из того же алфавита (соответственно группой символов того же размера и из того же алфавита). Узлы, реализующие эти преобразования, называются соответственно P-блоками (P-box, permutation box) и S-блоками (S-box, substitution box). В качестве примера приведём отечественный стандарт шифрования «Алгоритм криптографического преобразования ГОСТ 28147-89». Стандарт принят в СССР в 1989 г. Если охарактеризовать алгоритм ГОСТ в самом общем виде, – это блочный шифр, построенный по схеме Фейстеля с 32 циклами шифрования. Длина информационного блока – 64 бита, длина ключа – 256 бит. В текущем стандарте США на симметричную криптографию – шифре AES [20] – определена длина блока, равная 128 битам. Криптоалгоритм имеет переменную длин у блоков и различные длины ключей. Длина ключа и длина блока могут быть равны, независимо друг от друга, 128, 192 или 256 битам.

Поточные шифры почти всегда работают быстрее и требуют для своей реализации гораздо меньше программного кода, чем блочные шифры. Наиболее известный поточный шифр RC4 разработан Р. Ривестом; характеризуется переменным размером ключа и байт-ориентированными операциями. На один байт требуется от 8 до 16 действий, программная реализация шифра выполняется очень быстро. Независимые аналитики исследовали шифр. Он считается защищённым. RC4 используется для шифрования файлов в таких изделиях, как RSA SecurPC. Он применяется для защиты коммуникаций, например, для шифрования потока данных в Интернет-соединениях, использующих протокол SSL. Современная тенденция – использование блочных шифров в режиме поточного шифрования (например, поточное шифрование обеспечивают режимы CBF и OFB для алгоритма DES или режим гаммирования для алгоритма ГОСТ 28147-89).

К бесключевым криптосистемам относятся хеш-функции и генераторы псевдослучайных чисел. *Хеш-функции* – это математические или иные функции, принимающие на входе строку переменной длины (называемую прообразом) и преобразующие её в выводную сроку фиксированной (обычно меньшей) длины, называемую значением хеш-функции или сверткой [28]. Более формально определение представлено в [6]. Хеш-функции используются для контроля целостности сообщений, передаваемых по каналу связи, и генерации ЭЦП. Пусть $g:\{0,1\}^{n} \rightarrow \{0,1\}^{q(n)}$ – функция, вычислимая за полиномиальное от n время, $q(n)$ – некоторый полином. Такая функция называется *генератором*. Генератор g *псевдослучаен* [23], если порождаемые им последовательности неотличимы никаким полиномиальным вероятностным алгоритмом от случайных последовательностей той же длины $q(n)$. Качественный ключ, для использования в рамках симметричной криптосистемы, представляет собой случайный двоичный набор. Если требуется ключ разрядностью n, в процессе его генерации с одинаковой вероятностью должен получаться любой из 2^{n} возможных вариантов. Генерация ключей для асимметричных криптосистем – процедура более сложная, т.к. ключи, применяемые в таких системах, должны обладать определёнными математическими свойствами. Например, в случае системы RSA модуль шифрования представляет собой произведение двух больших простых чисел.

Классификация по стойкости криптоалгоритма

Способность криптосистемы противостоять атакам криптоаналитика называется стойкостью. Количественно стойкость измеряется как сложность наилучшего алгоритма, приводящего криптоаналитика к успеху с приемлемой вероятностью. Универсальный метод прямого перебора множества всех возможных ключей позволяет получить оценку сверху для стойкости алгоритма шифрования. Проблема всей современной криптографии – это отсутствие нижней границы стойкости; длина ключа задаёт лишь общий объём пространства ключей, но всегда есть вероятность, ткнув пальцем в небо, угадать решение. Относительное ожидаемое безопасное время определяется как полупроизведение числа открытых ключей и времени, необходимого криптоаналитику, чтобы испытывать каждый ключ [13]. В зависимости от целей и возможностей криптоаналитика меняется и стойкость. Различают стойкость ключа (сложность раскрытия ключа наилучшим известным алгоритмом), стойкость бесключевого чтения, имитостойкость (сложность навязывания ложной информации наилучшим известным алгоритмом) и вероятность навязывания ложной информации. Аналогично можно различать стойкость собственно криптоалгоритма, стойкость протокола, стойкость алгоритма генерации и распространения ключей [2].

В зависимости от сложности взлома алгоритмы обеспечивают различные степени защиты. Во главу угла ставится принципиальная возможность получения по перехвату некоторой информации об открытом тексте или использованном ключе. Существуют *безусловно стойкие* (или *теоретически стойкие*), *доказуемо стойкие* и *предположительно стойкие* криптоалгоритмы.

Теоретически стойкие системы создают шифртексты, с недостаточным количеством информации для однозначного определения соответствующих им текстов (или ключей). В лучшем случае открытый текст может быть локализован в достаточно большом подмножестве множества всех открытых текстов, и его можно лишь «угадать» с ничтожно малой вероятностью. Для совершенного шифра открытый текст «локализуется» во всём множестве открытых текстов. Тем самым, для него сама задача расшифрования становится бессмысленной. Никакой метод криптоанализа, включая полный перебор ключей, не позволяет определить ключ или открытый текст и получить некоторую информацию о них. Алгоритм, безусловно, стоек, если восстановление открытого текста невозможно при любом объёме шифртекста, полученного криптоаналитиком. Безопасность безусловно стойких криптоалгоритмов основана на доказанных теоремах о невозможности раскрытия ключа. Принципиально не раскрываемые шифры (например, совершенно секретные системы Клода Шеннона, в которых ключ не может использоваться повторно, а его размер больше либо равен объёму текста) неудобны на практике (симметричные криптосистемы с разовым использованием ключа требуют большой защищённой памяти для хранения ключей, системы квантовой криптографии требуют волоконно-оптических каналов связи и дóроги, а доказательство их безопасности уходит из области математики в область физики [2]). В силу своей непрактичности и высокой ресурсозатратности абсолютно стойкие шифры применяются только в сетях связи с небольшим объёмом передаваемой информации, когда можно обеспечить всех абонентов достаточным запасом случайных ключей и исключить возможность их повторного применения: обычно это сети для передачи особо важной государственной информации.

Стойкость *доказуемо стойких* криптоалгоритмов определяется сложностью решения хорошо известной математической задачи, которую пытались решить многие математики и которая считается общепризнанно сложной. В качестве примера можно привести системы DH (Диффи-Хеллмана) [21] и RSA (Ривеста-Шамира-Адельмана) [181, 19], основанные на сложностях дискретного логарифмирования и разложения целого числа на множители соответственно. Достоинство доказуемо стойких алгоритмов – хорошая изученность задач, положенных в их основу, а недостаток – невозможность в случае необходимости оперативной доработки криптоалгоритмов, т.е. отсутствие гибкости. Повышение стойкости достигается увеличением размера математической задачи или её заменой, что влечёт цепь изменений в аппаратуре, используемой для шифрования.

Предположительно стойкие криптоалгоритмы основаны на сложности решения частной математической задачи, которая не сводится к хорошо известным задачам. Примерами могут служить шифры ГОСТ 28147-89 [21], AES [20], FEAL [17]. Предположительно стойкие криптоалгоритмы характеризует сравнительно малая изученность математических задач, на которых базируется их стойкость. Такие шифры обладают большой гибкостью, что позволяет при обнаружении слабых мест не отказываться от алгоритмов, а дорабатывать их.

Получение оценок стойкости – довольно сложная задача [38]. Стойкость шифра рассматривается как разработчиком, так и критиком (криптоаналитиком). Оценки, заявленные разработчиком, могут быть опровергнуты, если кто-либо укажет способ

криптоанализа, для которого вычислительная сложность меньше заявленной.

Классификация по используемым средствам

Рассмотрим эту классификацию в применении к генераторам псевдослучайных чисел. Для генерации ключевой информации, предназначенной для использования в рамках симметричной криптосистемы, используются следующие методы (в порядке возрастания качества) [27]:

- ✦ программная генерация, предполагающая вычисление очередного псевдослучайного числа как функции текущего времени, последовательности символов, введённых пользователем, особенностей его клавиатурного почерка и т.д.;
- ✦ программная генерация, основанная на моделировании качественного псевдослучайного генератора с равномерным законом распределения;
- ✦ аппаратная генерация с использованием качественного псевдослучайного генератора;
- ✦ аппаратная генерация с использованием генераторов случайных последовательностей, построенных на основе физических генераторов шума и качественных псевдослучайных генераторов.

Лучший способ генерации множества случайных битов – извлечение их из естественно случайных событий реального мира [28]. Часто такой метод требует наличия специальной аппаратуры, но можно реализовать его и на компьютерах. Дж. Б. Эгнью предложил генератор случайных битов, который можно встроить в СБИС [31]. Это конденсатор металл-изолятор-полупроводник. Два таких конденсатора помещаются рядом друг с другом, а случайный бит определяется разностью зарядов этих конденсаторов. Другой генератор случайных битов генерирует поток случайных битов, используя нестабильность частоты свободно колеблющегося осциллятора [32. Коммерческая микросхема фирмы AT&T генерирует случайные числа, опираясь на это явление [30], а генератор М. Гьюда [29] собирает случайные числа из физических явлений (например, радиоактивного распада). В качестве случайных величин можно также рассматривать интервалы между нажатиями клавиш клавиатуры. Главный недостаток подобных систем – возможные закономерности в генерируемой последовательности. Используемые физические процессы могут быть случайными, однако использование измерительных инструментов может привести к появлению проблем: смещения, отклонения или корреляции между битами. Обойти эти недостатки можно, используя не один, а несколько случайных источников. В качестве случайных событий Брюс Шнайер предлагает рассматривать, например [28: нажатия на клавиши; команды мыши; номер сектора, время дня и задержку поиска для каждой дисковой операции; фактическое положение мыши; номер текущей строки развёртки монитора; содержимое текущего выводимого на экран изображения; содержимое таблиц файловой системы FAT; загрузку процессора; время поступления сетевых пакетов и т.д. Counterpane опубликовала новый класс атак на генераторы случайных чисел [Counterpane_], основанный на работе компании над коммерческими моделями. Одна из самых неожиданных находок – определённые генераторы случайных чисел могут быть надёжными при использовании с одной целью, но ненадёжными для другой; обобщение анализа надёжности опасно.

Классификация по наличию сертификата

Согласно действующему на территории России законодательству, если организация использует несертифицированные в России криптографические алгоритмы шифрования и ЭЦП данных, она не может вести обмен документами с государственными учреждениями. Обеспечить юридическую значимость электронных документов при обмене ими между пользователями позволит использование сертифицированных ФАПСИ криптоалгоритмов.

Классификация криптоаналитических атак

Современная криптография — соревнование методов шифрования и криптоанализа. Криптоанализом (от греч. kryptós – «скрытый» и analýein – «ослаблять» или «избавлять») называют науку восстановления (дешифрования) открытого текста без доступа к ключу. Попытка криптоанализа называется *атакой*. Разработанная и описанная ниже классификация позволяет различать криптоаналитические атаки и их последствия одновременно по нескольким параметрам:

- ✦ по доступу к открытому и зашифрованному тексту на основе:
  - ✧ только шифртекста;
  - ✧ открытого текста;
  - ✧ подобранного открытого текста;
  - ✧ адаптивно подобранного открытого текста;
  - ✧ информации из побочных каналов;
- ✦ по контролю над процессом:
  - ✧ пассивные;
  - ✧ активные;
    - ❖ имитация сообщения;
    - ❖ подмена сообщения;
    - ❖ по исходу атаки:
  - ✧ полный взлом;
  - ✧ глобальная дедукция;
  - ✧ частичная дедукция;

✧ информационная дедукция;
✦ по критическим ресурсам:
✧ память;
✧ время;
✧ данные;
✦ по степени применимости к различным шифрам:
✧ универсальные;
✧ для определенной категории шифров;
✧ для конкретного криптоалгоритма;
* по используемым средствам:
✧ математические методы;
✧ устройства-перехватчики физических параметров процесса шифрования;
✧ эволюционное программирование;
✧ квантовые компьютеры;
✦ по последствиям:
✧ нарушение конфиденциальности;
✧ нарушение целостности;
✧ нарушение доступности;
✦ по возможности распараллеливания:
✧ распределённые;
✧ не распределённые.

Классификация по доступу к открытому и зашифрованному тексту

Прежде чем классифицировать атаки, введём ряд обозначений: открытый текст обозначим буквой x, шифртекст – буквой y (в качестве x может выступать любая последовательность битов: текстовый файл, оцифрованный звук, точечный рисунок и т.д.). Пусть для зашифрования и расшифрования используются ключи k и k' соответственно (в симметричной криптографии $k = k'$); обозначим функцию зашифрования E_k, расшифрования – D_k. Тогда выполняются соотношения $E_k(x) = y$, $D_{k'}(y) = x$.

До недавнего времени по критерию доступа к открытому и зашифрованному тексту выделялось четыре основных типа криптоаналитических атак. В последнее время одним из самых актуальных направлений криптоанализа стало осуществление атак, использующих особенности реализации и рабочей среды. *Атаки по сторонним* (или *побочным, каналам*) – это вид криптографических атак, использующих информацию, полученную по сторонним или побочным каналам [Жуков_]. В каждом случае предполагается (согласно фундаментальному допущению Кирхгоффа), что криптоаналитик знает используемый алгоритм шифрования.

✦ ***Атака на основе только шифртекста.*** Криптоаналитик располагает шифртекстами $y_1,...,y_m$, полученными из неизвестных открытых текстов $x_1,...x_m$ различных сообщений. Требуется найти хотя бы один из x_i, $i = \overline{1,m}$(или соответствующий ключ k_i), исходя из достаточного числа m криптограмм, или убедиться в своей неспособности сделать это. В качестве частных случаев возможно совпадение ключей: $k_1 =...= k_m$ или совпадение открытых текстов: $x_1 =...= x_m$.

✦ ***Атака на основе открытого текста.*** Криптоаналитик располагает парами $(x_1,y_1),...,(x_m,y_m)$ открытых и соответствующим им зашифрованных текстов. Требуется определить ключ k_i для хотя бы одной из пар. В частном случае, когда $k_1 =...=k_m = k$, требуется определить ключ k или, убедившись в своей неспособности сделать это, определить открытый текст x_{m+1} ещё одной криптограммы y_{m+1}, зашифрованной на том же ключе.

✦ ***Атака на основе подобранного открытого текста*** отличается от предыдущей лишь тем, что криптоаналитик имеет возможность выбора открытых текстов $x_1,...,x_m$. Цель атаки та же, что и предыдущей. Подобная атака возможна, например, в случае, когда криптоаналитик имеет доступ к шифратору передающей стороны.

✦ ***Атака на основе адаптивно подобранного открытого текста.*** Это частный случай вышеописанной атаки с использованием подобранного открытого текста. Криптоаналитик может выбирать используемый шифруемый текст и уточнять свой последующий выбор на основе полученных ранее результатов шифрования.

✦ ***Атака на основе информации из побочных каналов.*** Криптоаналитик располагает информацией, которая может быть получена с устройства шифрования и не является при этом ни открытым текстом, ни шифротекстом.

Атаки по побочным каналам классифицируются [Жуков_]:

✦ по контролю над вычислительным процессом: *пассивные* и *активные*;
✦ по способу доступа к модулю: *агрессивные* (invasive), *полуагрессивные* (semi-invasive) и *неагрессивные* (non-invasive);
✦ по методу, применяемому в процессе анализа: *простые* – simple side channel attack (SSCA) и *разностные* – differential side channel attack (DSCA);
✦ по виду используемого побочного канала:
✧ атаки по времени исполнения (Timing Attacks);
✧ атаки по энергопотреблению (Power Analysis Attacks);

- ✧ атаки по ошибкам вычислений (Fault Attacks);
- ✧ атаки по электромагнитному излучению (ElectroMagnetic Analysis);
- ✧ атаки по ошибкам в канале связи (Error Message Attacks);
- ✧ атаки по кэш-памяти (Cache-based Attacks);
- ✧ акустические атаки (Acoustic Attacks);
- ✧ атаки по световому излучению (Visible Light Attacks).

Атаки с использованием известного или подобранного открытого текста встречаются чаще, чем можно подумать. Необходимое требование к хорошему криптоалгоритму – способность противостоять таким атакам. Это означает, что рассекречивание некоторой информации, передававшейся по каналу связи в зашифрованном виде, не должно приводить к рассекречиванию другой информации, зашифрованной на этом ключе. Указанное требование учитывает особенности эксплуатации аппаратуры и допускает некоторые вольности со стороны оператора или лиц, имеющих доступ к формированию засекреченной информации. В среде криптоаналитиков нельзя назвать неслыханными факты добычи открытого текста шифрованного сообщения или подкупа лица, которое должно будет зашифровать избранное сообщение. Применяются и «косвенные» методы осуществления атаки на основе подобранного шифротекста. Злоумышленник может убедить обладателя секретного ключа переслать некое сообщение, но в зашифрованной форме. В [36] описан приём, использованный командованием военно-морского флота США во время Второй мировой Войны перед битвой на Мидвее. Чтобы убедиться в правильности результатов работы по взлому японского военного шифра, криптоаналитики США попросили американский гарнизон, дислоцировавшийся на Мидвее, сообщить по открытому незащищённому каналу о нехватке пресной воды. Спустя два дня было перехвачено секретное сообщение, в котором японцы, осуществлявшие мониторинг использованного канала, сообщали о проблемах с водой в некоем «AF». Благодаря этому американцы узнали: «AF» – кодовое обозначение Мидвея в шифрограммах противника. Атаки на основе подобранных текстов считаются наиболее опасными.

Классификация по контролю над процессом шифрования

К классу *пассивных* атак [8] относят действия противника, который «пассивно изучает» шифрованные сообщения, может их перехватить и подвергнуть криптоанализу с целью получения информации об открытом тексте или ключе.

Современные технические средства позволяют потенциальному противнику «активно» вмешиваться в процесс передачи сообщения. Различают два типа активных атак: *имитация* и *подмена сообщения*.

Атака имитации: противник «вставляет» в канал связи сфабрикованное им «шифрованное сообщение», которое на самом деле не передавалось от законного отправителя к получателю. Противник рассчитывает, что получатель воспримет это сообщение как подлинное (аутентичное).

Атака подмены: противник, наблюдая передаваемое по каналу связи подлинное сообщение отправителя, «изымает» его и заменяет поддельным. Различные шифры могут быть более или менее уязвимыми к активным атакам.

Способность самого шифра (без использования дополнительных средств) противостоять активным атакам обычно называют имитостойкостью шифра. Количественной мерой имитостойкости шифра служат вероятности успеха имитации и подмены соответственно. Эти вероятности определяют шансы противника на успех при навязывании получателю ложного сообщения.

Классификация по исходу атаки

Криптоанализ ставит своей задачей в разных условиях получить дополнительные сведения о ключе шифрования, чтобы уменьшить диапазон вероятных ключей. Результаты криптоанализа могут варьироваться по степени практической применимости. Взлом шифра совсем не обязательно подразумевает обнаружение способа, применимого на практике для восстановления открытого текста по перехваченному зашифрованному сообщению. В научной криптологии другие правила [3]. Шифр считается взломанным, если в системе обнаружено слабое место, которое может быть использовано для более эффективного взлома, чем метод полного перебора ключей (‘brute-force approach’). Допустим, для дешифрования текста методом полного перебора требуется перебрать 2^{128} возможных ключей; тогда изобретение способа, требующего для дешифрования 2^{110} операций по подбору ключа, будет считаться взломом. Такие способы могут требовать нереалистично больших объёмов подобранного открытого текста или памяти ЭВМ. Под взломом понимается лишь подтверждение уязвимости криптоалгоритма, свидетельствующее о том, что свойства надёжности шифра не соответствуют заявленным характеристикам. Как правило, криптоанализ начинается с попыток взлома упрощённой модификации алгоритма, после чего результаты распространяются на полноценную версию: прежде чем браться за взлом, например,

16-раундовой версии DES, естественно для начала попытаться взломать шифр с меньшим количеством раундов, чем указано в его спецификации (например, 8-раундовую версию шифра).

Криптограф Ларс Кнудсен [7] предлагает следующую классификацию успешных исходов криптоанализа блочных шифров в зависимости от объёма и качества секретной информации, которую удалось получить:

- *полный взлом* – криптоаналитик извлекает секретный ключ;
- *глобальная дедукция* – криптоаналитик разрабатывает функциональный эквивалент исследуемого алгоритма, позволяющий зашифровывать и расшифровывать информацию без знания ключа;
- *частичная дедукция* – криптоаналитику удаётся расшифровать или зашифровать некоторые сообщения;
- *информационная дедукция* – криптоаналитик получает некоторую информацию об открытом тексте или ключе.

Классификация по объёму необходимых ресурсов

Атаки можно классифицировать по объёму ресурсов, необходимых для их осуществления:

- *память* – объём памяти, требуемый для реализации атаки;
- *время* – количество элементарных операций, необходимых выполнить;
- *данные* – необходимый объём открытых и соответствующим им зашифрованных текстов.

В некоторых случаях эти параметры взаимозависимы: например, за счёт увеличения памяти можно сократить время атаки.

Классификация по степени применимости к различным шифрам

Если цель криптоаналитика – раскрытие возможно большего числа шифров (независимо от того, хочет ли он этим нанести ущерб обществу, предупредить его о возможной опасности или просто получить известность), то наилучшая стратегия – разработка универсальных методов анализа [2, 6]. Эта задача наиболее сложная.

Любую задачу отыскания способа раскрытия некоторой конкретной криптосистемы можно переформулировать как привлекательную математическую задачу, при решении которой удаётся использовать многие методы той же теории сложности, теории чисел и алгебры. Это привело к появлению методов криптоанализа, применимых к различным классам шифров. Например, можно выделить группу методов, применимых для раскрытия асимметричных криптосистем. Эти методы, можно разделить по критерию эффективности их использования для взлома систем шифрования, основанных на сложности задачи дискретного логарифмирования или факторизации, и т.д.

Наконец, существуют атаки, использующие уязвимость при проектировании или реализации конкретного шифра. Эти атаки не могут быть в общем случае перенесены на целый класс шифров, однако могут эффективно применяться для взлома определённого криптоалгоритма. К этому классу атак относятся, например, атаки по побочным каналам.

Классификация по используемым средствам

Почти все осуществлённые на практике удачные атаки на криптосистемы используют слабости в реализации и размещении механизмов криптоалгоритма. Такие атаки основаны на корреляции между значениями *физических параметров, измеряемых в разные моменты во время вычислений* (потребление энергии, время вычислений, электромагнитное излучение и т.п.), и внутренним состоянием вычислительного устройства, имеющим отношение к секретному ключу. На практике атаки по побочным каналам на много порядков эффективнее, чем традиционные атаки, основанные только на математическом анализе. Атаки по побочным каналам используют особенности реализации (поэтому их иногда называют также атаками на реализацию – implementation attacks) для извлечения секретных параметров, задействованных в вычислениях. Такой подход менее обобщённый, поскольку привязан к конкретной реализации, но зачастую мощнее классического криптоанализа.

На данный момент методы, основанные на использовании новых информационных технологий (эволюционного программирования и квантовых компьютеров) в криптоанализе, не привели к сколько-нибудь серьёзным прорывам во взломе шифров и представляют в большей степени академический, чем практический интерес. Тем не менее, эти методы заслуживают внимания из-за своей оригинальности; не исключено, что со временем их значение в криптологии возрастёт.

Криптосистему можно рассматривать как «чёрный ящик», т.е. устройство или программу, чья структура не известна, но подавая сигналы команды или данные на её вход, можно получить реакцию на выходе. Задача криптоанализа – идентификация этой системы, т.е. определение её структуры на основе сигналов, поступающих на вход и получаемых на выходе. Инструментом решения этой задачи могут быть нейронные сети; её теория изложена в [35].

Брюс Шнайер в книге «Прикладная криптография» [28] склоняется к пессимизму в отношении применимости нейронных сетей в криптоанализе: «Процесс взлома не оставляет места обучению: вы либо раскрываете ключ, либо нет. (По крайней мере, это верно при вскрытии любого надёжного алгоритма). Нейронные сети хорошо работают в структурированных средах, допускающих обучение, но не в высокоэнтропийном, предположительно случайном мире криптографии». Тем не менее, исследования в этом направлении продолжаются.

Генетические алгоритмы успешно применяются в криптоанализе перестановочных и подстановочных шифров [4]. В статье [3] предлагается метод криптоанализа шифров на основе алгоритма укладки ранца, базирующийся на использовании генетических алгоритмов. Отличительная особенность такого подхода – его универсальность, т.е. возможность применения к любой версии «рюкзачной» криптосистемы, а также простота работы.

Главным достижением в области квантовых вычислительных алгоритмов стал алгоритм Шора разложения чисел на множители. На сегодняшний день есть конкретные результаты. Так, IBM продемонстрировала использование созданного в лабораториях компании семикубитового квантового компьютера для факторизации чисел по алгоритму Шора. Хотя решённая им задача вряд ли способна поразить воображение (компьютер верно определил, что делителями числа 15 являются числа 5 и 3), это самое сложное вычисление за всю историю квантовых компьютеров. В настоящее время не существует технических средств, позволяющих за полиномиальное время разложить достаточно большое число на множители. Таким образом, самым важным вопросом остаётся создание квантового компьютера. Алгоритм Шора чрезвычайно прост и довольствуется гораздо более скромным аппаратным обеспечением, чем то, которое понадобилось бы для универсального квантового компьютера. Вероятно, что квантовое устройство для разложения на множители будет построено задолго до того, как весь диапазон квантовых вычислений станет технологически осуществимым.

Классификация по последствиям атаки

Возможные последствия реализации атаки рассмотрим с точки зрения нарушения свойств информации – конфиденциальности, целостности и доступности. Выделим три стратегии действий, которые может предпринять нарушитель в случае успешной реализации атаки:

- ✦ перехват информации, передаваемой по каналам связи;
- ✦ модификация информации, передаваемой по каналам связи (подмена, ложные сообщения, блокирование передачи и т.д.);
- ✦ работа от чужого имени (обход средств аутентификации участников информационного взаимодействия по каналам связи).

Классификация по возможности распараллеливания

Распараллеливанию поддаются не все алгоритмы криптоанализа, однако оно позволяет значительно ускорить нахождение ключа. При оценке эффективности метода криптоанализа необходимо учитывать не только его временну́ю и ёмкостную сложность, но и возможность распараллеливания на многопроцессорной системе. Так, алгоритм Полларда [Ростовцев 98] имеет сложность $O(\sqrt{n})$, однако *не поддаётся распараллеливанию*. В то же время, метод полного перебора, который на однопроцессорной машине уступает по эффективности методу Полларда, – простой пример метода криптоанализа, *допускающего распараллеливание*.

Известно два направления в организации параллельного вычисления ключа [Амамия_].

Во-первых, построение конвейера. Представим алгоритм соотношения $E_k(x) = y$ в виде детерминированной цепочки простейших действий (операций): $O_1, O_2, ..., O_N$.

Возьмем N процессоров $A_1, A_2, ..., A_N$, зададим их порядок и положим, что i-ый процессор выполняет три одинаковые по времени операции:

1) приём данных от (i–1)-го процессора;
2) выполнение операции O_i;
3) передача данных следующему (i+1)-му процессору.

Тогда конвейер из N последовательно соединенных, параллельно и синхронно работающих процессоров работает со скоростью $v/3$, где v – скорость выполнения одной операции процессором.

Второе направление распараллеливания в том, что множество K разбивается на непересекающиеся подмножества $K_1, K_2, ..., K_Q$. Система из машин перебирает ключи так, что i-ая машина осуществляет перебор ключей из множества K_i, $_i = \overline{1,Q}$. Система прекращает работу, если одна из машин нашла ключ. Самая большая сложностью в изложенном подходе – организация деления ключевого множества. Однако если организовать поиск ключа так, чтобы при каждом очередном опробовании каждый из N процессоров стартует со случайной точки, время опробования увеличится, но схема значительно упростится. Как показано в работе [Грушо_], среднее число шагов опробования N процессорами (машинами) ключей из множества K в этом случае составляет $|K|/N$.

Реализация такого параллелизма предполагает различные решения. Самое очевидное решение – создание компьютерного вируса для распространения программы-взломщика в глобальной сети. Идея впервые опубликована в [34]. Вирус должен использовать периоды простоя компьютера (по данным исследований, компьютер простаивает 70–90% времени) для осуществления перебора по множеству ключей. Рано или поздно один из заражённых компьютеров обнаружит искомый ключ (необходимо предусмотреть механизм оповещения злоумышленника); с ростом производительности компьютеров и скорости распространения вирусов угроза успешного исхода такой атаки растёт.

В [12] описаны более оригинальные идеи распараллеливания вычислений: «Китайская лотерея», создание «криптоаналитических» водорослей и животных. Китайская лотерея предполагает: в каждый радиоприёмник и телевизор встроена микросхема, запрограммированная на автоматическую проверку различных множеств ключей после получения по эфиру пары открытый текст/шифртекст. Использование биотехнологий сделает возможным осуществление более эффективных атак. Рассмотрим вымышленное животное под названием «DESозавр». Оно состоит из оптически прозрачных биологических клеток, умеющих тестировать возможные ключи. По какому-то широковещательному оптическому каналу в клетки передаются пары открытый текст/шифртекст. Решения переносятся к органам речи DESозавра специальными клетками, путешествующими по кровеносной системе животного. В доисторические времена средний динозавр состоял примерно из 10^{14} клеток (без микробов). Если каждая клетка может выполнять миллион шифрований в секунду, вскрытие 56-битового ключа займет $7 \cdot 10^{-4}$ сек, а 64-битового – не более 0,2 сек. Другой биологический подход – создание методами генной инженерии криптоаналитических водорослей, умеющих вскрывать криптографические алгоритмы методом полного перебора. Водоросли могут покрывать большие пространства, что теоретически позволит создать некое подобие распределённого компьютера с огромным числом процессоров.

Классификация взломщиков криптосистем

При оценке стойкости криптосистемы необходимо учитывать возможности потенциального злоумышленника, который может осуществить атаки на систему. Иначе говоря, необходимо предварительно воссоздать собирательный образ (модель) нарушителя. Такая модель должна указывать [39]:

- ✦ категории лиц, в числе которых может оказаться нарушитель;
- ✦ предположения о квалификации нарушителя и его технической оснащённости;
- ✦ возможные цели нарушителя и ожидаемый характер его действий.

За основу для разработки классификационной схемы принята модель нарушителя, предложенная в [37]. Модель предназначалась для описания широкого класса нарушителей безопасности компьютерных систем, поэтому в параметры схемы внесены изменения с учётом того, что объект атак – криптосистема. Предложенная классификация позволяет при построении модели взломщика криптосистемы учитывать различные параметры и, тем самым, установить зависимость возможных сценариев атак от характеристик злоумышленников, со стороны которых система подвержена нападениям. В разработанной нами классификации злоумышленников можно различать по нижеперечисленным параметрам:

- ✦ по технической оснащённости:
  - ✧ персональный компьютер;
  - ✧ сеть ЭВМ;
  - ✧ суперкомпьютер;
- ✦ по конечной цели:
  - ✧ обнаружение слабости в алгоритме;
  - ✧ полный взлом алгоритма;
- ✦ по доступу к шифрующим средствам:
  - ✧ «внутренний» нарушитель;
  - ✧ «внешний» нарушитель;
- ✦ по уровню подготовки:
  - ✧ взаимодействие с компьютером на уровне пользователя;
  - ✧ математический аппарат;
  - ✧ программирование;
  - ✧ электротехника и физика;
  - ✧ социальная инженерия;
- ✦ по первичной информации о средстве шифрования:
  - ✧ пользователь;
  - ✧ криптограф;
  - ✧ «клептограф»;
- ✦ по возможности кооперации:
  - ✧ «одиночка»;
  - ✧ коллектив.

Классификация по технической оснащенности

Арсенал доступных средств для взлома шифров и конечные цели у хакера-любителя, коммерческой организации и учёного-криптоаналитика существенно различаются.

Крупной компании с большими вычислительными сетями под силу методом перебора вскрыть

ключ длиной 64–80 бит. Подтверждение тому – вскрытие RC5-64 (блочного шифра компании RSA, использующего 64-битный ключ. Стартовавший в 1997 г. на сайте www.distributed.net проект «распределённого взлома», в котором на добровольной основе приняли участие более 300 тысяч пользователей глобальной сети, успешно завершён за пять лет (1757 дней) – за это время перебрано 85% всего пространства ключей [http://www.distributed.net/rc5/_].

Для решения задач криптоанализа можно использовать и суперкомпьютеры. В статье [43] приведён список организаций, обладающих самыми мощными суперкомпьютерами и способных заниматься перехватом и дешифрованием информации; описана зависимость роста мощности суперкомпьютеров от времени, исходя из которой можно определить предположительную стойкость криптоалгоритмов.

Классификация по конечной цели

Действия хакеров связаны с целенаправленным преодолением (взломом) программно-аппаратных средств защиты для получения или модификации вполне определённых данных [39]. В основе их действий обычно корыстная мотивация, реже – желание прославиться, нанести моральный ущерб законному владельцу зашифрованной информации или иные причины.

Цель специалиста-криптоаналитика – создание новых и повышение эффективности существующих методов анализа стойкости криптографических средств. Каждый новый метод криптоанализа приводит к пересмотру безопасности шифров, к которым он применим.

Классификация по доступу к шифрующим средствам

Нарушителем может быть не только постороннее лицо, но и законный пользователь системы, а также лицо из числа обслуживающего персонала [39]. Если в качестве взломщиков выступают ненадёжные сотрудники компании, возможностей для осуществления атак существенно больше, чем у любых других взломщиков.

Классификация по уровню подготовки

Квалификация взломщика определяется наличием релевантных знаний, умений и навыков. Выделяют следующие основные области, освоение которых может быть полезным нарушителю для осуществления атаки на криптосистему:

- ✦ взаимодействие с компьютером на уровне пользователя – для осуществления атак с использованием доступных инструментальных средств;
- ✦ математический аппарат – для создания новых методов криптоанализа и повышение эффективности существующих методов;
- ✦ программирование – для разработки инструментальных средств, реализующих алгоритмы криптоанализа; создания вирусов для распараллеливания поисков ключа и т.п.;
- ✦ электротехника и физика – для реализации криптоатак по побочным каналам с использованием информации, которую можно извлечь из шифрующего устройства. Например, взломщик может отслеживать энергию, потребляемую смарт-картой, когда она выполняет операции с закрытым ключом, такие, как расшифрование или генерация подписи. Противник может замерять время, затрачиваемое на выполнение криптографической операции, или анализировать поведение криптографического устройства при возникновении определённых ошибок [9];
- ✦ социальная инженерия – мощное оружие взломщика, позволяющее обойти защиту самых стойких криптосистем, воспользовавшись доверчивостью пользователей.

Градация злоумышленников по их квалификации может быть различной. Например, может быть выделено три класса злоумышленников [41]:

- ✦ высококвалифицированный злоумышленник-профессионал;
- ✦ квалифицированный злоумышленник – не профессионал;
- ✦ неквалифицированный злоумышленник – не профессионал.

Классификация по первичной информации о средстве шифрования

Нарушителю может быть известна информация (в том числе секретная) о принципах функционирования криптосистемы. Так, одна из причин ненадёжности криптосистем, согласно [27], – использование слабых ключей. Слабый ключ – это ключ, не обеспечивающий достаточного уровня защиты или использующий в шифровании закономерности, которые могут быть взломаны. Это означает: если для генерации ключей используется криптографический слабый алгоритм, независимо от используемого шифра вся система будет нестойкой [28]. Генераторы случайных чисел – место, в котором часто ломаются криптографические системы. Зная принцип извлечения случайных чисел, злоумышленник может значительно сократить область перебора возможных ключей системы.

Особого внимания заслуживает технология, описанная в работе [45] под названием «клептография».

Разработчик может встроить в криптосистему лазейки, позволяющие получать доступ к зашифрованной информации без знания секретного ключа. Через «чёрный ход» информированный человек может легко преодолеть защиту. Если механизм действия шифра держится в секрете, вероятность наличия подобной «лазейки» повышается.

Классификация по возможности кооперации

В последнее время в связи с развитием сетей (в частности, Интернета), стало возможно эффективно использовать метод «грубой силы» (перебора) путём распараллеливания операций. Нередко профессиональные хакеры объединяются в преступные группировки, стремящиеся к наживе и выполняющие хищение конфиденциальной информации по заказам конкурирующих фирм и даже иностранных спецслужб [39. Альтернативный вариант – создание вируса, незаметно для пользователя устанавливающего на подключённый к сети компьютер программу, способную осуществлять дешифрование сообщения путём перебора ключей. Сценарий такой коллективной атаки прост и описан в [44]. После запуска программа подключается к серверу, получает от него набор ключей для перебора и после окончания работы возвращает результат. Программа может работать в фоновом режиме, в качестве скринсейвера или активироваться по ночам. Такой подход применим не только для взлома шифров, но и для подбора двух текстов, имеющих одинаковое значение хэш-функции, вычисленной указанным алгоритмом.

Заключение

Оценка эффективности криптографических средств защиты информации – сложная научно-техническая задача. При выборе криптосистемы необходимо проводить анализ угроз безопасности в конкретной компьютерной системе, предусматривающий оценку его стойкости к достаточно разнообразным типам криптоаналитических нападений. Для различных по назначению и принципам построения криптосистем, а также в зависимости от вида и ценности защищаемой информации, наиболее опасная модель поведения потенциального нарушителя может быть различной. Более важная информация обычно становится объектом нападения более квалифицированного и осведомлённого нарушителя.

Модель угроз можно рассматривать как композицию модели злоумышленника, модели атак и модели криптосистемы. Эти модели могут быть построены на основе классификаций, предложенных в данной работе. Они обладают свойствами расширяемости, формализуемости, простоты и многофакторности. Применение построенных классификаций для оценки эффективности используемых криптографических средств позволяет провести всесторонний анализ угроз, которым может подвергаться конкретная система в определённых обстоятельствах. Такой подход отличается гибкостью и позволяет в полной мере учитывать особенности криптографических средств и области их использования. ■

Литература

1. Амамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект. М.: Мир, 1993.
2. Ростовцев А.Г., Михайлова Н.В. Методы криптоанализа классических шифров // 1998. Опубликовано: http://crypto.hotbox.ru/download/cryptoan.zip
3. Spillman R. Cryptanalysis of knapsack ciphers using genetic algorithms // Cryptologia, 17(1), 1993.
4. Spillman R., Janssen M., Nelson B., Kepner M. Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers // Cryptologia. 17(1), 1993. P. 31-44.
5. Matthews R. The use of genetic algorithms in cryptanalysts // Cryptologia. 17(2), 1993. P. 187-201.
6. Грушо А.А., Применко Э.А., Тимонина Е.Е. Анализ и синтез криптоалгоритмов. Курс лекций. М.: 2000.
7. Knudsen L.R. Block Ciphers – Analysis, Design, Applicatons // Ph.D. dissertation, Aarhus Unversity, Nov 1994.
8. Зубов А.Ю. Криптографические методы защиты информации. Совершенные шифры: Учебное пособие. М.: Гелиос АРВ, 2005.
9. Жуков А.Е. Криптоанализ по побочным каналам (Side Channel Attacks). // Материалы конференции РусКрипто – 2006.
10. Kerckhoffs A. La cryptographie militaire // Journal des sciences militaires, vol. IX. P. 5-38, Jan. 1883, (P. 161–191, Feb. 1883).
11. Баричев С. Основной вопрос криптографии // Chief Inforamtion Officer – руководитель информационной службы. #5 (37), 2005, с. 93–95.
12. Quisquarter J.-J., Desmedt Y.G. Chinese Lotto as an Exhaustive Code-Breaking Machine // Computer, v.24, n.11, Nov 1991. P. 14–22.
13. Осипян В.О., Осипян К.В. Криптография в задачах и упражнениях. М.: Гелиос АРВ, 2004.
14. Oppliger R. Contemporary Cryptography. Artech House Publishers , 2005, 510 p.
15. Шеннон К.Э. Работы по теории информации и кибернетике. М.: И.Л., 1963.

16. Merkle R.C., Hellman M.E. Hiding Information and Signatures in Trapdoor Knapsacks // IEEE transactions on Information Theory. V. 24, n. 5, Sep 1978. P. 525–530.
17. Shimizu R., Miyaguchi S. Fast Data Encipherment Algorithm FEAL // Transactions of IEICE of Japan, v. J70-D, n. 7, Jul 87. P. 1413–1423.
18. Rivest R.L., Shamir A., Adleman L.M. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems // Communications of the ACM, v.21, n.2, Feb 1978. P. 120–126.
19. Rivest R.L., Shamir A., Adleman L.M. On Digital Signatures and Public Key Cryptosystems // MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212, Jan1979.
20. RIJNDAEL description. Submission to NIST by Joan Daemen, Vincent Rijmen // Available via http://csrc.nist.gov/encryption/aes/round1/docs.htm
21. Diffie W., Hellman M. New directions in cryptography // IEEE Trans. Inform. Theory, 22 (1976). P. 644–654.
22. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.
23. Введение в криптографию / Под общей ред. В.В. Ященко. СПб.: Питер, 2001
24. Brassard J. Modern Criptolog. Springer-Verlag, Berlin – Heidelberg, 1988. – 107 p. (Русский перевод: Брассар Ж. Современная криптология. Полимед, 1999. 176 стр.)
25. Shneier B. Snake Oil, Crypto-Gram // February, 1999. Available via http://www.counterpane.com/Crypto-Gram.html
26. Скляров Д.В.Искусство защиты и взлома информации. БХВ-Петербург 2004, 271с.
27. Иванов М.А. криптографические методы защиты информации в компьютерных системах и сетях // М.: КУДИЦ-ОБРАЗ, 2001.
28. Schneier B. Applied Cryptography Second Edition: protocols, algorithms and source code in C. John Wiley & Sons Inc., 1996. (Русский перевод: Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: ТРИУМФ, 2002.)
29. Gude M. Concept for a high-Performance Random Number Generator Based on Physical Random Phenomena // Frequenz, v. 39, 1985. P. 187–190.
30. AT&T. T7001 Random Number Generator // Data Sheet, Aug 1986
31. Agnew G.B. Random Sources for Cryptographic Systems // Advances in Cryptology – EUROCRYPT'87 Proceedings, Springler-Verlag, 1988. P. 77–81.
32. Fairfield R.C., Mrotenson R.L., Koulthart K.B. An LSI Random Number Generator (RNG) // Advances in Cryptology: Proceedings of CRYPTO 84, Springler-Verlag, 1985. P. 115–143.
33. Kelsey J., Schneier B., Wagner D., Hall C. Cryptanalytic Attacks on Pseudorandom Number Generators // Fast Software Encryption, Fifth International Workshop Proceedings (March 1998), Springer-Verlag, 1998, pp. 168–188. Available via: http://www.counterpane.com/pseudorandom_number.html
34. White S.R. Covert Distributed Processing with Computer Viruses. // Advances in Cryptology – CRYPTO'89 Proceedings, Springler-Verlag, 1990. P. 616–619.
35. Галушкин А. Теория нейронных сетей. М.:ИПРЖР, 2000.
36. Kahn D. The Codebreakers: The Story of Secret Writing. MacMillan, New York, 1967.
37. Степашкин М.В., Котенко И.В., Богданов В.С. Оценка защищенности компьютерных сетей на основе анализа графов атак // Пятая Общероссийская Конференция «Математика и безопасность информационных технологий» (МаБИТ-06). Москва, МГУ, 2006.
38. Молдовян Н. Каким быть новому стандарту шифрования? //»Компьютерра» №2 от 18.01.2000.
39. Дорошенко А.Н., Ткачев Л.Л. Информационная безопасность. Методы и средства защиты информации в компьютерных системах. Учебное пособие. М.: МГУПИ, 2006. – 143 с.
40. RIJNDAEL description. Submission to NIST by Joan Daemen, Vincent Rijmen. Available via: http://csrc.nist.gov/encryption/aes/round1/docs.htm
41. Корнюшин П.Н., Костерин А.С. Информационная безопасность: Учебное пособие. – Владивосток: ТИДОТ ДВГУ, 2003. – 154 с
42. [http://www.distributed.net/rc5/_] http://www.distributed.net/rc5/
43. http://www.ssl.stu.neva.ru/psw/crypto/Pudov21.html
44. Тульчинский А. Взлом криптоалгоритмов: мифы и реалии // «Компьютерра» №12 от 19 мая 2003 г.
45. Жуков А.Е. Криптография и клептография Криптосистемы со встроенными лазейками // Опубликовано: http://www.ruscrypto.ru/netcat_files/File/seminar.005.zip, 25.05.2004.

ТЕСТИРОВАНИЕ КРУПНЫХ КОМПЛЕКСОВ ПРОГРАММ НА СООТВЕТСТВИЕ ТРЕБОВАНИЯМ

В.В. Липаев,
главный научный сотрудник Института системного программирования РАН, профессор Государственного университета – Высшей школы экономики
algordom@mtu-net.ru

Рассмотрены проблемы разработки корректных требований к крупным системам и комплексам программ; тестирование и реализация требований к крупным программным продуктам.

Концепция тестирования программных продуктов на соответствие требованиям

Крупные комплексы программ – компоненты вычислительных систем, реализующие обычно их *основные функциональные свойства и задачи*, и создающие предпосылки для последующего их развития и применения. Реализация жизненного цикла, методология управления, обработки информации и изменения программных комплексов зависят от квалификации специалистов, технических, организационных и договорных требований заказчика и от сложности проекта. Множество текущих состояний и модификаций компонентов в жизненном цикле крупных комплексов программ, обусловливают необходимость упорядочивать, контролировать их развитие и реализацию участниками проекта. Организованное, контролируемое и методичное отслеживание создания и динамики изменений в жизненном цикле комплексов программ, их разработка при строгом учёте и контроле каждого изменения – базовоя проблема эффективного, поступательного развития крупных систем высокого качества *методами программной инженерии* [7, 8, 10]. Накопление в мире знаний, опыта разработки и применения огромного количества различных сложных комплексов и компонентов программ для ЭВМ способствует систематизации и обобщению методов и технологий их разработки, сокращению дефектов и неопределённостей в характеристиках и качестве поставляемых и применяемых программных продуктов.

Методология программной инженерии основана на применении систематизированного, научного и предсказуемого процесса планирования, проектирования, разработки, тестирования и сопровождения крупных программных продуктов. Разработчики таких продуктов должны учитывать, что они работают в организационных и финансовых рамках заключённых контрактов, и искать решение поставленной перед ними задачи с учётом требований и условий заказчика. Программная инженерия *не рассматривает технические аспекты индивидуального создания (программирования) модулей и небольших компонентов*, в её ведении *задачи коллективов специалистов* по управлению проектами и разработке методов, теорий и средств, необходимых для обеспечения жизненного цикла крупных комплексов программ.

Особенности тестирования крупных программных комплексов на соответствие требованиям заказчика:

- *объектами тестирования* должны быть крупные, целостные комплексы программ (сотни тысяч строк) для систем обработки информации и управления в реальном времени, разрабатываемые большими коллективами (командами) специалистов;
- программные продукты должны соответствовать *первому эталону* – целостному *комплексу требований* к функциям, характеристикам, архитектуре и качеству, утверждённому заказчиком и согласованному с разработчиком;
- комплексы применяемых тестов рассматриваются как *второй эталон* функций и характеристик программного средства, которые должны

адекватно отражать и покрывать набор требований заказчика к программному продукту;

- ✧ разработка требований и тестирование комплексов программ осуществляется методами программной инженерии в условиях *регулярного планирования, управления и ограниченных ресурсов*, выделяемых заказчиком;
- ✧ применяются *сценарии и процедуры* детерминированных тестов и *генераторы динамических,* стохастических тестов на основе аттестованных, сложных моделей внешней среды и систем реального времени;
- ✧ процессы и продукты формирования требований и реализации тестирования должны быть охвачены глубоким, детальным *документированием, архивированием и конфигурационным управлением* исходными, промежуточными и отчётными документами;
- ✧ в крупных программных комплексах применяются *апробированные программные компоненты и модули высокого качества.*

Программные продукты всё чаще *встраиваются* в различные сложные системы реального времени. Работа над такими проектами требует от программных инженеров и тестировщиков, широкого взгляда и освоения *общих проблем проектирования и применения систем* определённого назначения и сферы использования. Программному инженеру необходимо участвовать в разработке требований для всей системы, а также освоить прикладную область применения создаваемого комплекса программ до начала тестирования и обдумывания функций компонентов, их характеристик и тестов, требованиям которых должен будет отвечать программный продукт.

Проблема целевого управления крупными проектами систем и программных продуктов включает рациональное использование и предупреждение потери ресурсов путём сбалансированного распределения их по частным работам и компонентам на протяжении всего жизненного цикла объекта.

Управление крупными проектами – особый вид инженерной деятельности, включающий: постановку задач; подготовку требований; планирование, организацию и стимулирование специалистов; контроль хода работ и использования ограниченных ресурсов при создании сложных систем. Критический параметр управления такими проектами – *ограниченное время.* Основная проблема целевого управления работами – сводить воедино усилия исполнителей-специалистов разной квалификации, подрядчиков и субподрядчиков, добиваясь, чтобы они выступали при создании компонентов и всей системы как *команда*, а не как разрозненная группа независимых, функциональных специалистов. В результате должны обеспечиваться *концептуальная целостность системы* и высокое качество решения её главных целей и функций *при сбалансированном использовании ресурсов* на реализацию всей совокупности задач.

Проблему рационального сочетания целей, стратегий действий, конкретных процедур и доступных ресурсов необходимо решать для достижения *основной цели получения программного продукта с заданными требованиями и качеством* [3, 5]. Планирование проектов крупных комплексов программ должно *обеспечивать компромисс* между требующимися функциями и характеристиками создаваемой системы, и ограниченными ресурсами, необходимыми на её разработку и применение.

В жизненном цикле любых крупных систем и комплексов программ *четыре ключевые объекта* определяют их назначение, создание, свойства и качество:

- ✧ требования заказчика и/или пользователей к системе, определяющие цели создания и назначение программного продукта, его функции и область применения, необходимые пользователям системы и иным заинтересованным лицам, в заданной эксплуатационной среде;
- ✧ *тесты-эталоны и вторая адекватная форма описания содержания и функционирования программного продукта* для сквозной проверки полноты реализации требований и верификации соответствия исходным требованиям к системе, комплекса и компонентов программ, способствующие исключению дефектов и ошибок при выполнении и изменении требований;
- ✧ *эксплуатационная документация,* обеспечивающая применение программного продукта пользователями в соответствии с требованиями к его функциями и характеристикам, и являющаяся *третьим адекватным эталоном* корректности их реализации;
- ✧ *программный продукт и система – результат реализации требований:* функциональные характеристики системы, её поведение и предусмотренные свойства, реализованные в системе и комплексе программ для обеспечения корректного применения; критические параметры, характеристики качества и эффективности; экономические характеристики производства программного продукта и эксплуатационные риски.

Эти четыре объекта ниже используются как базовые при анализе тестирования в жизненном цикле крупных комплексов программ. Основное внимание сосредоточено на исследовании и формализации полноты и качества:

- *разработки требований* к функциям и характеристикам крупных программных продуктов;
- *тестирования и обеспечения корректности реализации* этих требований.

Основная *цель тестирования комплексов программ и их функциональных компонентов* – обнаружить, зарегистрировать и устранить дефекты и ошибки, которые внесены во время последовательной *разработки и реализации требований* к функциям и характеристикам комплекса программ. Тестированию программных компонентов и модулей посвящена обширная литература, однако в ней мало внимания уделяется тестированию крупных, целостных комплексов программ на соответствие исходным требованиям к их функциям. В статье *акцент изменён* – всё содержание ориентировано на тестирование крупных программных комплексов, а тестирование компонентов и модулей не рассматривается; считается, что они применяются готовыми, апробированными и высокого качества [1, 2, 11]. Однако допускается, что для идентификации, локализации и устранения дефектов и ошибок необходимо применять тестирование и корректировку модулей и небольших компонентов с использованием соответствующих традиционных методов и средств.

Формализация требований к программному продукту (первый эталон)

В *требованиях к программному продукту* должно быть зафиксировано соглашение между заказчиком и выполняющими проект специалистами, отражающее потребности заказчика и пользователей в таком виде, чтобы разработчик мог построить удовлетворяющий их комплекс программ и его компоненты [3, 5]. Требования являются *исходным эталоном* при разработке любой системы или программы, должны быть достаточно конкретными, чтобы при тестировании и испытаниях можно было установить, когда они удовлетворены. Вся разработка проекта должна вытекать из требований, а все спецификации программного комплекса и компонентов найти отражение в процессах и продуктах разработки. Эти данные должны устанавливать состав, содержание и значения результатов исполнения программ, получаемыми системой или пользователями при определённых условиях и исходных данных. Любое отклонение результатов функционирования программы от предъявляемых к ней требований и сформированных по ним эталонов следует квалифицировать как дефект или ошибку в программном продукте и по возможности устранять.

При разработке требований к крупным системам и программным продуктам необходимо учитывать функции, основные свойства и общие требования к проектам крупных систем [3, 5, 6]. Формирование назначения, функций и технического задания на проект системы должно включать в себя основные требования к комплексу программ и особенности требований к нему заинтересованных лиц. Общие требования к качеству функционирования крупных программных продуктов реального времени обязательно включают: требования к надёжности и функциональной безопасности продуктов; требования к производительности и эффективности использования ресурсов ЭВМ программными продуктами в реальном времени, а также к допустимым рискам при их применении. Архитектурные требования к комплексам программ должны предусматривать управление изменениями требований, организацию изменений и сопровождение требований, обеспечение повторного использования программных компонентов и комплексов. Для обеспечения качества программного продукта необходима верификация и трассирование требований к функциям и компонентам комплекса, а также обеспечение баланса требований к ним. Должны быть обеспечены процессы достаточно полного и корректного документирования требований к функциям и характеристикам комплексов программ.

При формировании требований к системе и программному продукту заказчик и разработчики должны осуществлять согласованные действия в соответствии с принятой политикой и процедурами [7, 9, 10]:

- определить функциональные границы системы в терминах её поведения и предусмотренных свойств;
- определить каждую функцию, которая должна быть реализована в системе и программном продукте для обеспечения корректного применения;
- определить необходимые ограничения по реализации, обусловленные требованиями заказчика или неизбежными ограничениями системы, связанными с принятыми решениями;
- определить технические и потребительские характеристики, позволяющие оценивать технические результаты;

- задать системные требования и функции, в соответствии с которыми определяются риски и параметры системы и программного продукта, связанные с критическими показателями, надёжностью, безопасностью, производительностью;
- проанализировать целостность системных требований для обеспечения уверенности в том, что каждое требование и/или набор требований обладают системной целостностью;
- продемонстрировать связь между требованиями и практическими потребностями заказчика;
- на протяжении всего жизненного цикла вести учёт состояния совокупности требований вместе с их обоснованиями, связанными решениями и допущениями.

В требованиях должно устанавливаться *ожидаемое поведение системы и программного продукта,* выраженное, желательно, количественными характеристиками, а также границы их допустимых значений. Следует определить, насколько хорошо система, включая операторов-пользователей, приспособлена для реализации заданных функций. Требования могут зависеть от абстрактных представлений о подходящих характеристиках системы и включать методы и виды моделирования функционирования и среды, для достаточно полного описания заданных требований. Критические показатели функционирования анализируются и проверяются для подтверждения удовлетворения требований заказчика и для определения стоимости проекта, проектных графиков или эксплуатационных рисков, описывающих процессы установления, определения и использования соответствующих характеристик. Каждое положение *требований к системе и программному продукту должно проверяться* для установления его уникальности, полноты, непротиворечивости, совместимости с другими требованиями, реализуемости и проверяемости при испытаниях. Требуется удостоверять, что требования, с одной стороны, *необходимы и достаточны* для удовлетворения заказчика, а с другой – необходимые и достаточные входные данные для других процессов, например, *для проектирования функций и архитектуры комплекса программ.*

При формировании требований следует учитывать, что *источниками ошибок* в комплексах программ являются специалисты – конкретные люди с их индивидуальными особенностями, квалификацией, талантом и опытом. Можно выделить предсказуемые дефекты требований, модификаций, расширения и совершенствования комплекса, и необходимые изменения, обусловленные выявлением случайных, непредсказуемых дефектов и ошибок [7, 9]. Поэтому плотность потоков и размеры необходимых *корректировок в требованиях* к комплексу и компонентам программ могут различаться в десятки раз. Однако в крупных комплексах программ статистика и распределение типов ошибок и выполняемых изменений для коллективов разных специалистов нивелируются и проявляются достаточно общие закономерности, могущие использоваться как ориентиры при их выявлении. Каждому типу необходимых корректировок соответствует более или менее определённая категория специалистов – источники дефектов данного типа. Эту корреляцию целесообразно учитывать как *общую качественную тенденцию* при анализе и поиске их причин. Этому могут помогать оценки типовых дефектов и корректировок путём их накопления и обобщения по опыту разработки определённых классов комплексов программ в конкретных предприятиях.

Одна из основных причин ошибок в комплексах программ – *организационные дефекты при определении и использовании требований к программному продукту,* отличающиеся от остальных типов и условно выделяемые как самостоятельные. Ошибки и дефекты данного типа появляются из-за недостаточного понимания коллективом специалистов целей и функций комплекса программ, а также вследствие отсутствия чёткой его организации и поэтапного контроля требований качества продуктов. Это порождается *пренебрежением руководителей к организации всего технологического процесса* формализации требований сложных программных продуктов и приводит к серьёзной недооценке их дефектов, трудоёмкости и сложности их выявления и устранения. Для сокращения этого типа массовых ошибок, активную роль должны играть лидеры – менеджеры и аналитики-системотехники, способные вести контроль и конфигурационное управление требованиями, изменениями и развитием версий и компонентов комплексов программ.

Формализация тестирования программного продукта (второй эталон)

Совокупности требований к тестам могут применяться *как эталоны и вторая адекватная форма описания функций и характеристик комплексов программ.* Для обеспечения высокого качества программных продуктов стратегию создания *требований к тестам* целесообразно строить, *основываясь на требованиях к функциям и характеристикам* крупных комплексов программ. Требования к тестам можно дополнять структурным методом на основе анализа логики архитектуры системы. Он может

применяться в зависимости от условий договора или особых требований к безопасности или ограниченным рискам системы. Например, покрытие полной структуры решений для некоторых функций может быть необходимо в аэрокосмических и других критических системах с повышенными требованиями безопасности.

Тестирование комплексов программ может иметь *две цели,* одинаково важные для качества тестирования. Первая цель – тестирование для идентификации дефектов – подразумевает успешность процедуры тестирования, если *дефект найден и устранён.* Это отличается от подхода в тестировании (вторая цель), когда тесты исполняются для демонстрации того, что программный комплекс удовлетворяет предъявляемым требованиями и, соответственно, тест считается успешным, если *не найдено дефектов.* Тестирование программ может использоваться для демонстрации наличия дефектов, но никогда не гарантирует их отсутствие. Основная причина этого в том, что *полное, всеобъемлющее тестирование недостижимо* для реального крупного программного продукта [1, 4, 9]. Когда пытаются оценивать методы тестирования, надо чётко определять, что подразумевается под их эффективностью, желательно, в количественных величинах. Только обладая такого рода данными можно говорить о корректности сравнения и оценки эффективности разных методов тестирования.

При разработке сложных комплексов программ надо иметь пригодные для применения тестирования *эталоны – требования и/или сценарии использования функций* программного продукта. Их необходимо анализировать и определять в терминах адекватных спецификаций требований к функциям и характеристикам программного продукта, а также к тестам, устанавливающим их содержание и корректность. Вторая форма описаний содержания программного продукта должна служить для сквозной верификации спецификаций требований к тестам сверху вниз, и подвергаться верификации на корректность соответствия исходным требованиям к комплексу и компонентам программ разного уровня.

Для обеспечения высокого качества программного продукта параллельно с *верификацией требований* и разработкой корректировок, следует разрабатывать и верифицировать *спецификации и сценарии тестов,* отражающие методы и конкретные процедуры проверки реализации этих требований. Тестовые спецификации могут использоваться для проверки согласованности, внутренней непротиворечивости и полноты реализации требований без исполнения программ. Для каждого требования к функциям комплекса программ, его архитектуре, функциональным компонентам должна быть разработана спецификация требований к тестам, обеспечивающая проверку корректности, адекватности и возможности в последующем реализовать тестирование компонента на соответствие этому требованию. Взаимная проверка корректировок функций компонентов, отражённых в требованиях и в спецификациях тестов, обеспечивает повышение их качества, сокращение дефектов, ошибок, неоднозначностей и противоречий.

Спецификации тестов должны обеспечивать дополнительный контроль корректности спецификаций требований и верификацию взаимодействия компонентов на соответствующем уровне описания комплекса программ. Независимая разработка спецификаций тестов на основе спецификаций требований, создаёт базу для обнаружения не тестированных требований или требований, которые принципиально не могут быть проверены тестированием. Таким образом, *верификация спецификаций требований тестов* к функциям и характеристикам программных компонентов и комплекса могут использоваться с *двумя целями:*

- ✧ для разработки и проверки программ и интерфейсов взаимодействия программных компонентов разных уровней в комплексе программ;
- ✧ для создания требований к скоординированному комплексу тестов для проверки совокупности компонентов, обеспечивающих взаимную проверку реализации спецификаций требований комплексом программ.

Такие *параллельные взаимные проверки* спецификаций требований и текстов программ, и спецификаций тестов способствуют выявлению и исключению множества вторичных дефектов и ошибок комплексов программ. Впоследствии эти спецификации тестов должны использоваться для непосредственного тестирования исполнения требований к программным компонентам соответствующего уровня. Параллельная и независимая разработка спецификаций программ и спецификаций тестов, а также их реализация позволяет распараллеливать работы, что ведёт к сокращению сроков создания компонентов и комплексов программ требуемого качества.

Реализация этих целей верификации и тестирования может производиться разными методами и *независимыми специалистами* – программистами, интеграторами и тестировщиками. Это позволяет использовать результаты их деятельности для сравнения

одних и тех же описаний программ, представленных на языках программирования и на языках тестов. Особенности описаний и реализации программ, и *мышление их разработчиков (программистов)* – на основе требований, функций, характеристик структуры и исполнения программ – существенно *отличаются от представлений и методов описаний тех же функций комплекса программ тестировщиками* – создателями сценариев и эталонов требуемых результатов тестирования. Они акцентируют свою деятельность на конкретных процедурах проверки функционирования, возможных результатах и взаимодействии компонентов комплекса программ. Это позволяет выявлять вторичные дефекты, и повышать качество *сопоставлением двух методов* и результатов описания *одних и тех же функций и характеристик программ* за счёт того, что мала вероятность одинаковых ошибок в сценариях и реализациях тестов и в описаниях требований к функциям и характеристикам программ.

Тестирование на базе верифицированных требований к тестам должно обеспечивать качество комплексов программ. Такие сценарии предназначены для поэтапной проверки внутренней непротиворечивости и полноты реализации требований к функциям и характеристикам качества комплексов программ. Требования к тестам должны проверяться на корректность системных взаимосвязей функциональных компонентов *сверху вниз* и на внутреннюю корректность взаимодействия между требованиями для компонентов и их пригодности для последующего тестирования и применения.

Тестовое покрытие – это отображение пространства варьирования тестов на некоторую совокупность требований к функциям и характеристикам компонентов или комплекса программ, предназначенных для тестирования. Покрытие можно отражать *с помощью матрицы,* где по одной оси представлены тестовые сценарии, а по другой – требуемые функции, характеристики, безопасность или риски компонентов или комплекса программ [4, 6, 12]. При этом можно отслеживать степень покрытия требований тестами, поддерживаемыми конфигурацией и сценариями использования программного продукта. Такие документы – статические *графовые модели* функционирования программного продукта. Подобные модели могут фиксировать потоки и виды данных системы, последовательность состояний, в которых может находиться программный продукт и последовательность их обработки. Можно проводить оценку степени покрытия относительно этих моделей, проверяя, что каждому узлу и каждой связи в графе соответствует тестовый сценарий. Оценка покрытия требует учёта, в какой степени тесты нацелены на пространство требуемых функций и характеристик комплекса программ. Точность определения степени покрытия ограничена, но выделение *сильных и слабых связей между тестами и пространством требований* к функциям и характеристикам продукта позволяет акцентировать анализ на безопасности, допустимых рисках или некоторых характеристиках, особенно важных для конкретного проекта.

Для обеспечения эффективности затрат ресурсов, крупномасштабное тестирование должно быть интегрировано как можно раньше с основными процессами проектирования, разработки и сопровождения в жизненном цикле программных комплексов, должно проводиться, начиная от анализа общих требований, заданных в техническом задании и/или в спецификации на всю систему, до детальных испытаний программных продуктов и их взаимодействия с внешней средой. По существу, требования к комплексу программ и тесты для проверки их адекватности и полноты, *отражают один и тот же объект,* но в разной форме. Поэтому сложность представительного описания комплекса тестов *соизмерима* со сложностью описания полной совокупности требований к функциям и характеристикам соответствующего комплекса программ. Вследствие этого *трудоёмкость и другие ресурсы* для разработки адекватных тестов *должны соответствовать ресурсам и трудоёмкости* при создании и реализации корректных требований, определяющих полностью программный продукт. Однако на практике это обычно не учитывается, и выделяемые на тестирование ресурсы оказываются значительно меньше и недостаточными для полноценного тестирования требований к крупным комплексам программ, что определяет их неполное соответствие требованиям и недостаточное качество.

Жизненный цикл разработки тестов и процессов тестирования программных комплексов должен проходить во времени, *параллельно динамике изменения требований* и жизненному циклу реализации комплекса программ. Тесты и сценарии их применения должны быть адекватными и полностью отражать содержание и изменения функций и характеристик компонентов комплексов программ, но в иной форме. Их следует рассматривать и анализировать как ещё одну форму описания изменений функций комплексов программ которые могут содержать дефекты и ошибки. Таким образом, программной (процедурной) форме представления программ должно полностью соответствовать его равноправное содержание в форме сценариев тестов

для проверки их взаимного соответствия. При этом *дефекты и ошибки возможны в обеих формах интегрального описания комплекса программ* (не только в текстах программного средства, но и в требованиях к тестам) определение их места и устранение – основная задача тестирования требований и тестов.

Квалификационное тестирование и испытания программного продукта на соответствие требованиям должно базироваться на организации, программе и методиках испытаний крупных программных продуктов. Тестирование на соответствие требованиям к динамическим характеристикам и рискам крупных программных продуктов должно включать тестирование надёжности и функциональной безопасности, характеристик производительности и динамического использования ресурсов ЭВМ, сокращение и ликвидацию опасных рисков при применении программных продуктов в реальном времени. Для обеспечения качества продукта важно тестирование эксплуатационной документации на соответствие требованиям к продукту. Тестирование должно поддерживать методы, процессы и средства управления конфигурацией требований и тестов крупных комплексов программ. Особое значение могут иметь процессы завершения испытаний и внедрения версий продуктов, анализ результатов, усовершенствование процессов тестирования требований и сертификация крупных программных продуктов.

Тестирование всегда предполагает компромисс между ограниченными ресурсами и заданными сроками, с одной стороны, и практически неограниченными требованиями к качеству результатов тестирования и программного продукта, с другой. *Оценка стоимости и затрат* и другие измерения процессов, связанных с оценкой ресурсов, необходимых для тестирования, как и оценка *экономической эффективности тестирования* на разных этапах и уровнях, основывается на точке зрения и практике менеджмента проектов и используется для оценки и улучшения процесса тестирования. Разная техника, концепции и модели тестирования требуют разных затрат по времени и необходимым ресурсам. Учёт соответствия между стоимостью или затратами, необходимыми для того или иного метода, и эффективностью тестирования – обязательная часть современного управления проектами крупных программных комплексов.

Характеристики качества встроенных программных продуктов реального времени зависят не только от их внутренних свойств, но и *от свойств внешней среды – сценариев и генераторов динамических тестов,* в которых они испытываются и применяются. Для сокращения неопределённостей и прямых ошибок при оценивании качества комплексов программ необходимо предварительно определить основные параметры внешней среды, при которых должен функционировать комплекс программ с требуемыми характеристиками. Для этого заказчик и разработчик совместно должны структурировать, описать и *согласовать модель внешней среды* и её параметры в среднем, типовом режиме применения программного продукта, а также в наиболее вероятных и критических режимах генерации тестов с обеспечением требуемых характеристик качества функционирования. Необходимость проверять широкий набор требований и характеристик крупных программных комплексов приводит к необходимости *динамического генерирования тестов* посредством создания моделей внешней среды, охватывающих весь спектр данных, воздействующих на тестируемый объект. Подготовку тестов для испытаний крупных комплексов программ целесообразно осуществлять с применением автоматизированной генерации динамических тестов внешней среды в реальном времени, а также средств обработки результатов динамического тестирования.

Формализация эксплуатационной документации программных продуктов (третий эталон)

Эксплуатационная документация должна обеспечивать *эффективное применение программного продукта* и точно отражать его назначение, функции, характеристики и требования, для использования квалифицированными специалистами-пользователями [7, 8, 10]. Для этого эксплуатационные документы необходимо тестировать на полное соответствие выполнения всей *совокупности требований на программный продукт,* согласованной между разработчиками и заказчиком. В результате эксплуатационные документы можно использовать как отдельный, независимый при разработке *третий эталон и вид тестов* для квалификационного тестирования реализации требований к функциям и характеристикам программного продукта. Апробирование документов пользователями при применении комплекса программ – *практический метод тестирования* корректности реализации требований к программному продукту, завершающий полный цикл испытаний и контроля его качества. Разработчики этих документов должны обеспечивать комфортное и корректное применение программных продуктов пользователями на основе ясного и непротиворечивого изложения в документах технологических процедур и операций для штатного применения, функционирования и получения требуемых результатов.

Организация документирования должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения эксплуатационных документов на программный продукт. Для этого должны быть выделены *специалисты,* обязанные планировать, утверждать, выпускать, распространять и сопровождать комплекты апробированных и утверждённых эксплуатационных документов. Они должны стимулировать разработчиков программных средств, осуществлять непрерывное, регламентированное документирование процессов и результатов своей деятельности, контролировать полноту и качество утверждённых эксплуатационных и отчётных документов.

Состав и содержание комплекта документов конкретного программного продукта, целесообразно адаптировать разработчиками к его особенностям и свойствам *на основе использования стандартов и типовых структур-шаблонов* для двух классов продуктов, в наибольшей степени различающихся особенностями эксплуатации [8, 10]. *Первый класс* составляют программные продукты автоматизированного управления динамическими объектами и процессами в реальном масштабе времени. В процессе их применения допускается минимальное вмешательство пользователями в их процедуры управления, и необходим небольшой объём эксплуатационных документов, выделяемых из стандартизированного, технологического комплекта. Для продуктов *второго класса* (для административных систем) возможно применение пользователями широкого набора процедур управления, регламентированных достаточно полным набором и подробным содержанием документов.

Документация администрирования при эксплуатации систем и крупных программных продуктов должна обеспечивать поддержку первичной инсталляции, безопасного функционирования и восстановления программ и данных после сбоев. Администратор системы и программного продукта должен быть информирован обо всех изменениях функционирования устройств системы и внешней среды, могущих привести к сбою или возникновению аварийной ситуации, и предпринимать соответствующие профилактические действия. Для этого требуется полная информация о *требованиях к программному продукту,* компонентам системы и внешней среды. Каждый из документов административного управления программным продуктом *должен не противоречить международным стандартам* на коммуникацию, интерфейсы с пользователями и базами данных, на защиту и обеспечение безопасности информации.

Документация для оперативных пользователей программных продуктов в системе, их функционирования, обработки и анализа результатов должна обеспечивать взаимодействие пользователей с различными аппаратно-программными реализациями терминалов. Для этого необходима унификация концепции, архитектуры, функций и методов визуализации пользовательского интерфейса. Типовые *формы-шаблоны документов* и процедуры работы с ними, рассматриваемые *как объекты стандартизации,* относятся к функциональному, оперативному уровню взаимодействия пользователей с системами и программными продуктами в соответствии с исходными требованиями к ним.

Программы для вычислительных машин – исключительно *гибкие изделия,* что отражается на массовом, не формализованном, «художественном» создании относительно небольших программных компонентов с нерегламентированными качеством и документами. Однако программные продукты и системы, поставляемые заказчикам или на рынок должны иметь *конкретные, гарантированные и документированные цели, функции и характеристики*. В то же время в жизненном цикле крупных комплексов программ невозможно в начале проекта предусмотреть все требования к функциям, характеристикам и качеству абсолютно точно, без дефектов и ошибок. Крупные программные продукты обычно имеют длительный жизненный цикл с рядом версий; они совершенствуются, расширяются, повышается их качество. Поэтому процессы формирования изменений и реализации *требований, тестов и эксплуатационных документов должны организовываться на основе методологии регулируемых итераций множества версий,* в которых периоды поставляемых, стабильных, испытанных версий программных продуктов, чередуются с интервалами пошагового расширения функций, совершенствования качества и оперативного адаптирования к характеристикам внешней среды у пользователей или в системах. Для этого в крупных проектах целесообразно применять международные стандарты, регламентирующие жизненный цикл комплексов программ, и *жёсткую дисциплину* координируемой деятельности коллектива специалистов для сопровождения и документируемого *конфигурационного управления версиями* требований, тестов и документов для обеспечения качества реализации изменяющихся требований.

Заключение

В мире накоплено огромное количество готовых программных компонентов, сокращается необходимость программирования и тестирования новых

модулей и повторно используемых компонентов. Это приводит к *увеличению роли интеграции* готовых компонентов, применения соответствующих методов и инструментария программной инженерии. Однако вследствие их принципиальной новизны и сложности, они трудно воспринимаются традиционными программистами и тестировщиками модулей и небольших компонентов, и множеством преподавателей отечественной высшей школы. *Коренные отличия* между методами и инструментарием индивидуального, «художественного» программирования и тестирования небольших программ и технологией планомерной, регламентированной, инженерной разработки и тестирования крупных программных продуктов приводят к тому, что последние медленно осваиваются и входят в практику работы больших коллективов специалистов.

В стране быстро возрастает, как общий объём производства программных продуктов, так и сложность отдельных проектов. Необходимость обеспечения взаимодействия распределённой вычислительной и коммуникационной техники предъявляет новые системотехнические требования к специалистам по программным комплексам. Это одна из причин возникновения проблем при разработке крупных программных продуктов, как и то, что многие предприятия, занимающиеся их производством, не уделяют должного внимания экономически эффективному применению современных методов управления проектами и международным стандартам, автоматизации и обеспечению всего жизненного цикла крупных программных комплексов. Эти проблемы *отражаются на экономике, качестве и конкурентоспособности отечественных программных продуктов,* на них целесообразно обращать внимание и решать заказчикам, руководителям и разработчикам крупных программных продуктов. ■

Литература

1. Бейзер Б. Тестирование черного ящика. Технология функционального тестирования программного обеспечения и систем. Пер. с англ. – СПб: ПИТЕР. 2004.
2. Блэк Р. Ключевые процессы тестирования. Пер. с англ. – М: ЛОРИ. 2006.
3. Вигерс К.И. Разработка требований к программному обеспечению. Пер. с англ. – М.: Русская редакция. 2004.
4. Дастин Э., Рэшка Д., Пол Д. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. Пер. с англ. – М. ЛОРИ. 2003.
5. Леффингуэлл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. Пер. с англ. – М.: Вильямс. 2002.
6. Липаев В.В. Отладка сложных программ. М.: Энергоатомиздат. 1993.
7. Липаев В.В. Программная инженерия. Методологические основы. Учебник. – М.: ТЕИС. 2006.
8. Соммервилл И. Инженерия программного обеспечения. 6-е издание. Пер. с англ. – М.: Вильямс. 2002.
9. Тэллес М., Хсих Ю. Наука отладки. Пер. с англ. – М.: Кудиц-образ. 2003.
10. Фатрелл Р. Т., Шафер Д. Ф., Шафер Л. И. Управление программными проектами: достижение оптимального качества при минимальных затратах. Пер. с англ. – М.: Вильямс. 2003.
11. Beizer B. Software testing techniques. N.Y.: Van Nostrand Reinhold. 1990.
12. Kit E. Software Testing in the Real World. NY. Addison-Wesley. 1996.

СИСТЕМНАЯ ДИНАМИКА: МОДЕЛИРОВАНИЕ ПРИНЯТИЯ СТРАТЕГИЧЕСКИХ И ОПЕРАТИВНЫХ РЕШЕНИЙ (метод функции капитального класса)

А.Р. Горбунов,
старший научный сотрудник Института США и Канады РАН
a_gorbunov07@mail.ru

Рассмотрено введение параметров и приёмов, облегчающих моделирование принятия решений в рамках методов системной динамики и имитационного моделирования, и предназначенных для построения алгоритмов и систем поддержки принятия решений, для разработки прогнозных сценариев больших индустриальных и экономических систем. Введены базовые алгебраические конструкции, улучшающие и развивающие логику моделей системной динамики и имитационного моделирования в области принятия решений.

Введение. Постановка задачи

В качестве широкой и универсальной научно-прикладной доктрины системная динамика предоставляет чрезвычайно разнообразные возможности рассмотрения задач принятия решений в различных аспектах стратегического управления [1, 2]. Наибольший интерес представляют методы, позволяющие анализировать существо основных альтернативных решений, обеспечивающие тонкое шкалирование параметров операций и их внешних условий. Характеристики стратегических выборов и альтернативных стратегий здесь как бы помещаются в систему технико-экономических и структурных координат, что облегчает их понимание, обобщение, дальнейшие расчёты и моделирование. И здесь чрезвычайно актуальны методы системной динамики, позволяющие формализовать широкие понятия, ввести новые оценочные категории, «овеществить» метод мышления эксперта и управленца. Они наиболее эффективны для нечётких и плохо формализуемых задач о принятии оптимальных решений и выбора оптимальной стратегии. В системно-динамических моделях общая картина операций фирмы рассматривается на фоне шкалы времени, что открывает путь для применения сценарного подхода к стратегическому управлению фирмой.

В дальнейших аналитических построениях вводятся контуры обратных связей, позволяющие сделать дополнительные заключения об аналитических свойствах модели и самого моделируемого объекта, перейти к углублённому анализу сценариев и прогнозов. Это делается в тесной привязке к предметной области, что облегчает взаимодействие аналитиков и менеджеров. Упомянутые особенности представляют особую ценность на предпроектном этапе (т.н. «фазе 0»), на этапе общего исследования проблемы, постановки задачи и оценки путей её решения. Данный подход наиболее эффективен при разработке концепции и конфигурации систем поддержки принятия решений (СППР), программировании и настройке их структурных модулей.

Анализируя способы и условия применимости методов системной динамики, мы избрали стратегию поиска упрощённых алгебраических аналогов, близких по смыслу графическим множителям Дж. Форрестера. Подобные алгебраические «конструкты» применяются в комбинации и взаимодействии

с графическими множителями Форрестера, обеспечивая тестирование, обоснование и развитие логики системно-динамических моделей. В результате заметно растут возможности моделирования, поскольку алгебраические множители допускают «подключение» эндогенных переменных и алгоритмов различной степени сложности.

«Средняя цена рынка» и капитальный класс продукции

Наибольший интерес представляют подходы, позволяющие выявить основное существо принимаемых решений на уровне ключевых аспектов особенностей бизнеса фирмы и оценки стратегической эффективности тех или иных решений и альтернатив.

В качестве стратегического параметра, разделяющего альтернативные стратегии и формирующие шкалы операционных выборов и альтернативных сценариев, применён параметр средней цены рынка данного продукта («average market price»). После его обработки при конструировании систем принятия он обобщён как капитальный класс продукции («capital class»). Эта категория шире рыночного класса («market class») или ценового класса («price class») [3], обычно рассматривается для продукции данного физического вида и в отличие от них способна обобщать выборы, связанные со стратегическим позиционированием фирмы, формированием производственно-сбытовой политики в целом, межотраслевыми решениями концернов.

Капитальный класс – стратегическая характеристика, обобщающая широкую панораму стратегических выборов не только в пределах данной группы продукции, но смежных и межотраслевых альтернатив концерна.

Диаграмма 1

Сферы применения категории капитального класса:

- ✦ продуктовые приоритеты и оптимальные структуры в рамках однородной ассортимента;
- ✦ продуктовые приоритеты и оптимизация разнородных производственно-сбытовых программ и портфелей бизнеса концернов;
- ✦ обоснование модификации продукции различной степени глубины, включая радикальные модификации базовых платформ и видов продукции;
- ✦ межотраслевые комбинации и трансформации бизнеса концернов и индустриальных систем.

Категория ценового (капитального) класса продукции сопряжена с базовым представлением о степени серийности продукции, когда относительно капиталоёмкая продукция обычно является и относительно дорогой, а крупносерийная – дешёвой и относительно массовой. Параметры серийности (масштаба) вводятся в операционные шкалы стратегических выборов. Происходит известный процесс «фокусировки стратегии» в виде группы значений технико-экономических показателей и параметров операций фирмы, отражающих её стратегическое позиционирование.

Практически любой объект, производственная система, включая услуги и сервисные операции, может быть «локализован» в шкалах удельного масштаба производства (средней серийности) и удельной капиталоёмкости (средней цены, капиталоёмкости продукции или операции). По общему правилу, объекты (товары, операции) с данной мерой капиталоёмкости имеют свою типичную меру масштаба и серийности производства (реализации). Однако возможны непредсказуемые и экстраординарные комбинации, обычно свидетельствующие о выходе на рынок гибридной или парадоксальной парадигмы бизнеса или о появлении технологического прорыва.

Цена и ценовой (капитальный) класс имеют одну и ту же «стоимостную» размерность. В прикладном смысле величина капитального класса может быть отождествлена с серединой ценового диапазона товара данного класса. Это «средняя цена товара со средними характеристиками в средних условиях». В некоторых случаях капитальный класс исчисляется исходя из технометрических исследований диаграмм «цена – характеристики» изделия. На его величину оказывают дизайнерские стилистические и иные «брендовые» особенности товара. В целом оценка средней цены рынка или ценового класса как стратегического параметра чувствительна к методу его оценивания. Специалисты-маркетологи способны достаточно убедительно интерпретировать эту величину на основе своего опыта в ходе отбора наилучшего шаблона управленческой модели [1]. Будучи измерителем удельной капиталоёмкости, ценовой капитальный класс нередко сопряжён с понятием «размерный класс» (dimension class) изделия или объекта, его массо-габаритными характеристиками. В связке цена – ценовой (капитальный) класс последний определяется не столько возможной ценой, сколько структурными характеристиками изделия, сопоставленными со средней ценой основных аналогов и конкурентов.

Категория капитального класса (средней цены, ценового класса) в различных формах вводится в функции и алгоритмы поиска оптимальных или рациональных решений, как оперативного, так и стратегического уровня. Она обусловливает оперативные решения по оптимизации продуктовой линейки (модельных рядов) и структуры производственно-сбытовых программ, стратегические решения по общему облику и позиционированию фирмы, глубокой модификации продукции, кардинально меняющей статус и облик товара, охватывает межотраслевые решения концернов и промышленных систем.

Принципиально альтернативные сценарии операций можно разместить на шкалах стратегических решений. Подобные модели способны находить операционный оптимум, группы наиболее вероятных и эффективных стратегий и решений (продуктов и технологий). В то же время они указывают область нестандартных решений и новых парадигм, комбинированных платформ бизнеса. Важная особенность этого подхода – рассмотрение в единой системе координат стратегических альтернатив и их операционных характеристик.

При таком обобщённом подходе стратегические выборы оцениваются по совокупности параметров экономической эффективности с учётом приоритетов и долгосрочных целей организации (стратегическая эффективность), для хозяйственных систем страны класса «концерн», «группа концернов», «отрасль промышленности», «промышленный комплекс» . Он обеспечивает теоретическую базу, общую логику СППР и управление сценарными расчётами при выборе стратегических приоритетов и структуры хозяйственного портфеляорганизации в рамках конкретных настроек СППР.

Функция поиска решений с фактором капитального класса (capital class function)

Применение моделей системной динамики потребовало новых подходов для анализа структурных зависимостей и процессов, лежащих в основе обоснования стратегических и операционных решений. В общем виде задача заключается во введении шкал структурной характеристики продукции и её специфического количественного ранга в модели технико-экономического и аналитического характера. В системах поддержки принятия решений и методов сценарного моделирования алгоритмы и алгебраические формулировки поиска решений условно обозначены как функции поиска решений с фактором капитального класса (capital class function), которая выступает как естественный и очевидный измеритель процессов роста добавленной стоимости и соответствующих базовых стратегий экономического развития.

На операционном и меганомическом уровнях (промежуточном между микро- и макроэкономическим) данная категория вводится в модели продуктовых выборов предприятий и концернов. Осуществляется комбинирование и оптимизация продуктовых программ, поскольку перебор ведётся по товарам и бизнес-портфелям – рациональным комбинациям продуктов, услуг и параметров их выпуска.

Категория средней цены рынка введена нами в системные модели принятия решений несколькими различными способами. В общем виде ситуация зависимости сбыта определяется ценой и ценовым классом.

Категория средней цены рынка вводилась в системные модели принятия решений несколькими различными способами. Рассмотрим степенной вариант записи:

$$Yq = f(P_b, P). \qquad (1)$$

Для этого рассмотрим как основной степенной вариант записи

$$Yq = \frac{f(x)}{P_b} \cdot \left(\frac{P_b}{P}\right)^m, \qquad (2)$$

где $f(x)$ – некоторая внешняя для данной модели величина ёмкости рынка;

Yq – натуральныйный или физический объём продаж в единицах продукции;

Pb – средняя цена рынка или мера ценового («капитального») класса;

P – исследуемая или проверяемая цена

m – «коэффициент упругости» или эластичность фактора капитального класса.

Модель позволяет изучить и измерить фактор капитального класса в различных системах и процессах. Точнее, в рассмотрение включается пары (Pb, P), символизирующие различные производственно-сбытовые стратегии с перебором не только по цене, но и по физическому облику продукции, выраженному через параметр капитального класса. Модель иллюстрирует интуитивно ясное представление о соответствии цены капитальному классу и его последствиях. При соответствии цены капитальному классу одну из них можно исключить. Опережение цены капитального класса

тормозит сбыт продукции; отставание – стимулирует его.

Обращает на себя внимание, что *m* выражает нелинейность второго порядка, вызванную несоответствием цены капитальному классу. Если несоответствие исчезает, исчезает и нелинейность второго порядка. Остаётся только нелинейность первого $f(x)/P$, а *m* можно рассматривать как меру ассиметрии факторов цены и капитального класса, поскольку эффект цены и капитального класса как явление начинает проявляться при *M*<>(неравно)1 и пропорциональна её величине.

Эту общую идею иллюстрирует запись:

$$Yq = \frac{f(x)}{P} + \left[\frac{f(x)}{P_b} \cdot \left(\frac{P_b}{P} \right)^m - \frac{f(x)}{P} \right]. \qquad (3)$$

Её смысл обусловлен тем , что

$$A+[B-A], \qquad (4)$$

где [*B*–*A*] – поправка на несоответствие цены капитальному классу.

Экономический смысл этой схемы в том, что она вводит представление о поправке к некоторой базовой величине вызванной несоответствием цены ценовому классу.

❖ При $m = 0$ цена (или капитальный класс) значения не имеет, рост капитального класса означает однозначно снижение продаж (отрицательный вклад).

❖ При $m = 1$ капитальный класс значения не имеет, рост цены однозначно означает снижение продаж.

❖ При $0 < m < 1$ – «от нуля до единицы» – роль капитального класса растёт (отрицательный вклад), «повышается» до нуля.

Ситуация $m = 1$ эквивалентна $P = Pb$. В целом это означает, что мы можем изменять траекторию *Yq* в меру *m*.

Возможен и несколько иной подход.

$$Yq = \left[\frac{f^2(x)}{P \times Pb} \cdot \left(\frac{Pb^{\,m}}{P} \right) \right]^{1/2} \qquad (5)$$

При $m = 1$ $\quad Yq = \dfrac{f(x)}{P}$ – капитальный класс «значения не имеет».

При $m = 0$ $\quad Yq = \dfrac{f(x)}{(P \times Pb)^{1/2}}$.

Здесь цена и капитальный класс имеют равный или симметричный (удельный) вес.

Можно вычислять поправку, вызванную *m* и несоответствием цены капитальному классу по схеме. Параметр *m* здесь «управляет» реакцией продаж на цены капитальному классу и её динамикой. Эта зависимость имеет свои «особые точки» при $m = 0$, $P = Pb$ со своим смыслом.

«Право на существование» имеет параметр «эластичности замещения» цены капитальным классом – при данных условиях одного и того же эффекта можно добиться изменением цены или переходом к другому структурному классу.

Модель является базовой или начальной для большого семейства различных прикладных и теоретических вариантов. Принята негативная реакция продаж на рост цены и капитального класса. Однако в реальной сложной системе стратегической конкуренции и сегментации рынка она носит неравновесный и нестандартный характер. Объём продаж даже крупного производителя при выборе рамочных параметров данной продуктовой стратегии зависит от цены и ценового класса продукции. Подобные алгоритмические конструкции определялись нами как функции соответствия (adequity function), поскольку они сфокусированы на соответствии цены капитальному классу в широком смысле адекватности стратегии, т.е. взаимному соответствию стратегических и операционных характеристик деятельности общим внутренним и внешним условиям её реализации, возможностям и шансам (strategic adequity).

Для соотношения цена – капитальный класс рассматривался также и экспоненциальный вариант записи вида

$$Yq = f(x)e^{-P\alpha}, \qquad (6)$$

где α – некоторый коэффициент, имеющий размерность обратной цены.

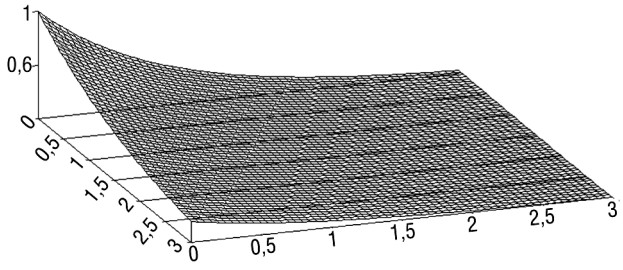
Исходя из логики формул (1 – 4) *Pb* тоже имеет размерность цены, поэтому чтобы разместить всё в одной формуле, можно записать:

$$Yq = f(x)e^{P\alpha - P_b\beta}. \qquad (7)$$

Выражение (7) есть функция 2-х переменных (*P, Pb*), примерный вид которой показан на *диаграмме 2.*

Функция (7) удобнее для чисто математической обработки, однако с нашей точки зрения, коэффициенты утрачивают интуитивную ясность и экономический смысл. В частности, параметры «альфа»

Диаграмма 2



и «бета» удобны математически, но их труднее интерпретировать с экономической точки зрения, чем, например, параметр m в (1–5), который обобщает степень и характер реакции на расхождение цены и капитального класса. Обобщающий параметр m в (6) можно ввести через $m(P-Pb)$, тем не менее, общий вид степенной функции выглядит предпочтительнее и ближе соответствует экономической логике. В то же время трудности математической интерпретации в имитационных моделях компенсируются алгоритмическими средствами.

В конечном счёте, предпочтение отдано степенной версии, поскольку органическая связь капитальной функции и производственных функций проявляется именно в её степенном варианте. В этом варианте она легко логорифмируется и переводится в процентную меру, что роднит её с производственными функциями классических версий, легко вводится в построение с моделями производственных процессов. Вклад фактора капитального класса удобно представлен как $(Pb/P)^m$.

Базовая модель производственной функции Кобба-Дугласа и множество её модификаций – степенные функции. Этот подход вводит в производственную систему новый фактор – капитальный класс, характеризующий физический облик выпуска. На данном этапе нами рассматриваются аналитические конструкции, предназначенные для применения в моделях системной динамики и имитационного моделирования.

Функция капитального класса в комбинации с методами производственного анализа[1] отражает процесс наращивания капиталоёмкости и капиталоинтенсивности производственной системы, что символизирует стратегию роста меры добавленной стоимости (Value Adding Strategy). В концепциях макроэкономического равновесия широко применяется категория усреднённого уровня цен, которая может быть дополнена усреднённым структурным параметром – средним капитальным классом выпуска. Динамика этой величины участвует в процессах циклического развития, структурного равновесия современной экономики. Это вносит новый аспект в проблемы, неоднократно изучавшиеся в рамках доктрин системной динамики [7,8].

Приведённые выше простейшие алгебраические конструкции имитационного (системно-динамического) моделирования призваны воспроизвести при помощи простых моделей основную логику исследуемых процессов, обеспечить обоснование и верификацию системных моделей.

Диаграмма 3

Решение задач с применением метода функции капитального класса

- задачи принятия решений по выбору оптимальных характеристик и технико-экономических платформ бизнеса. Стратегии ценообразования;
- задачи по оптимизации структуры производства, линеек продуктовых и модельных рядов по шкалам обобщённых технико-экономических характеристик (business portfolio). Ассортиментная политика и оптимизация хозяйственного портфеля. CVP-анализ;
- задачи макроэкономической и структурной динамики рассматривающие параметр удельной капиталоёмкости и физического (натурального) масштаба производства. Анализ производительности и стратегий роста меры добавленной стоимости, анализ конъюнктурных циклов и измерение технологического прогресса.

Системно-динамические подходы и генерические стратегии

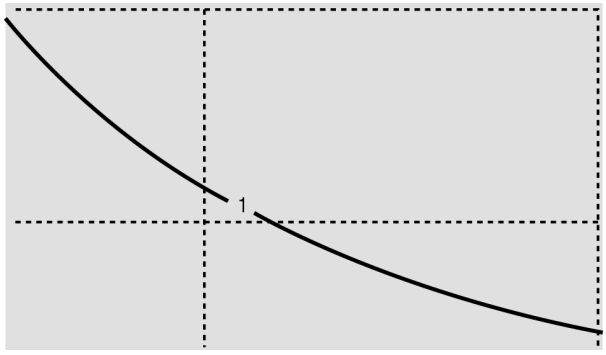
В данном подходе «эксплуатируются» аспекты системной динамики, обеспечивающие возможность концептуализации и анализа наиболее актуальных управленческих проблем. Функции, связанные с капитальным классом, вводятся через графические множители Форрестера. Они выражают экспертное знание проблемы и позволяют протестировать зависимости, существующие в различной, в том числе в трудно формализуемой и нерегулярной форме.

[1]Напр. Productivity Growth and U.S. Competitiveness, Edited by William J.Baumol and Kenneth McLennan? Oxfor University Press, 1985. (см. гл. 5 И.М. Осадчая, Е.З. Демиденко «Научно-технический прогресс: проблема количественных оценок) [2].

В классическом виде (1–5) продажи плавно убывают пропорционально цене.

Диаграмма 4

Равновесный вид ценовой функции



Сбыт продукции вовсе необязательно зависит от цены пропорционально. Одно из особых преимуществ системной динамики в том, что она позволяет рассматривать нестандартные и парадоксальные стратегии. Например, цена на продукцию класса «люкс» не может безболезненно занижаться, поскольку именно она в маркетинговом смысле сигнализирует об особом статусе изделия и особых условиях её продаж.

Для подобных ситуаций мы вводили специальный алгебраический аналог, символизирующий о ситуации, когда сбыт падает при любом несоответствии цены капитальному классу – и при его опережении, и при отставании.

$$Yq = f(Pb) \times \left[1 - \left(1 - \frac{Pb}{P}\right)^m\right]. \qquad (8)$$

где $f(Pb)$ – базовая мера «физического» объёма рынка для товара капитального класса $f(Pb)$.

Выражение $(1-[(1-Pb/P)]^m$ представляет собой системно-динамический множитель несоответствия цены капитальному классу. Обращает на себя внимание, что окрестности максимума при $m > 1$ имеют столовидный сглаженный вид (*диаграмма 5*).

При $0 < m < 1$ фактор капитального класса имеет вид пика (*диаграмма 6*).

На диаграмме 7 показана неравновесная особенность графика сбытовой функции, вызванная приведённым выше фактором несоответствия цены капитальному классу.

В выражении $Yq = [f(x)/Pb]\times[1-(Pb/P)^m]$ (8) комбинируются две ситуации. Первая заключается в том, что сбыт равновесно сокращается пропорционально капитальном классу (*диаграмма 4*). На неё «накладывается» вторая ситуация – пикообразная

Диаграмма 5

Сглаженная окрестность максимума фактора капитального класса

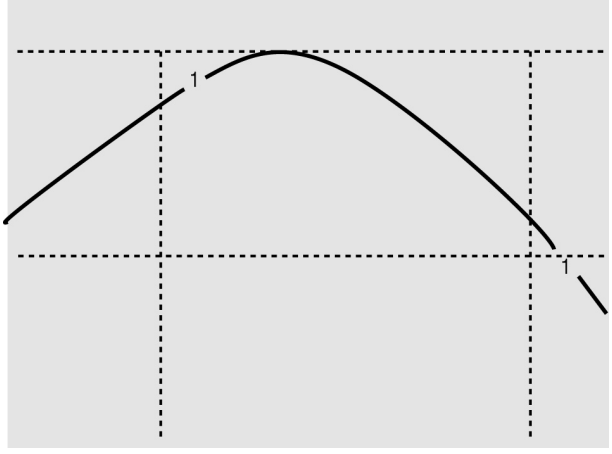


Диаграмма 6

Пикообразный вид максимума множителя фактора капитального класса неравновесного вида

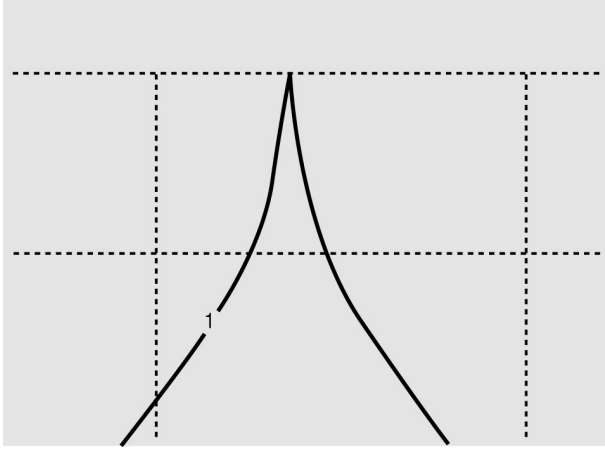
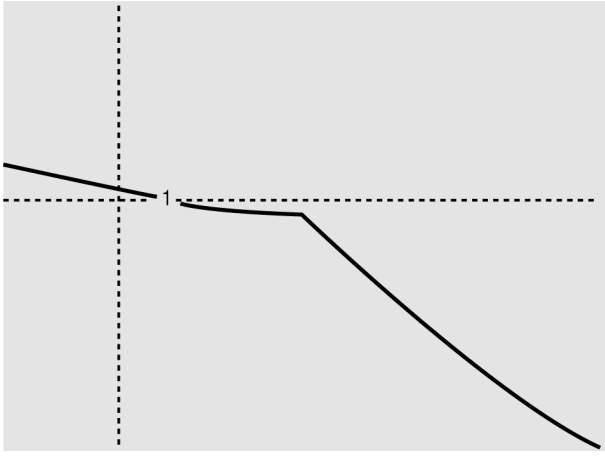


Диаграмма 7

Неравновесная область графика сбытовой функции с неравновесным фактором капитального класса



реакция рынка на несоответствие цены капитальному классу в окрестности точки *X* (*диаграмма 6*).

Подобная неравновесная зависимость может иметь произвольный ассиметричный вид, который достигается введением дополнительных параметров и множителей. Очевидно, параметр капитального класса неразрывно связан с производственным бюджетом *B* для масштаба производства *Yq*. Компания может добиться выигрыша в стратегии $Pb > P$ при преимуществах в издержках производства с учётом фактора производственной серии. Стратегия $P > Pb$ обычно означает возможность добиться от рынка признания цены, превышающей среднюю цену на товары данного капитального класса, что обычно обеспечивается стилистическими, дизайнерскими или имиджевыми факторами. Такую стратегию можно кратко охарактеризовать $P > Pb > B$, поскольку все величины имеют одну и ту же размерность.

При аналитической обработке стратегических решений мы применили краткую запись, символизирующую определённый сценарий операций компании

$$\text{strat}(Yq,\ Pb,\ P, B\ , Y...), \qquad (9)$$

где *Y* – доход или любое выражение интегрального хозяйственного результата;

q – стоимостное выражение продаж (прибыли) фирмы;

B – издержки производства количества продукции *Yq*;

P – цена продукции класса Pb;

Pb – структурный класс продукции выраженный как средняя цена рынка или мера удельной капиталоёмкости выпуска (капитальный класс).

В стратегию могут включаться другие параметры, всё более детализирующее описание операций компании. В рамках процедур обоснования решений могут рассматриваться кластерные группы стратегий (стратегические хозяйственные портфели), выборы, переходы между ними вида

$$\text{strat}(Yq_n,\ Pb_n,\ P_n, B_n\ , Y_n...)k. \qquad (10)$$

Приведённая выше схема и концепция функции капитального класса представляют собой обобщённую функцию стратегического соответствия («strategic adequity») продукции и стратегии фирмы по цене и структурному классу текущего и перспективного состоянию рынка. Она во многом выражает главное существо и обоснованность деловой стратегии фирмы, её бизнес-платформы. Упрощённые функции, в том числе множители Форрестера, символизирующие интуитивно воспринимаемое представление о соответствии параметров объективным внешним и внутренним условиям, объединены в семейство функций соответствия (adequity functions). Они не относятся к категории целевых функций (fitness functions) и интерпретируют, скорее, содержательные условия базовых моделей, подлежащих перебору и настройке. Они обычно предусматривают расхождения и «несоответствия» параметров в рамках определённой концепции или платформы бизнеса.

Модели принятия решений с фактором капитального класса

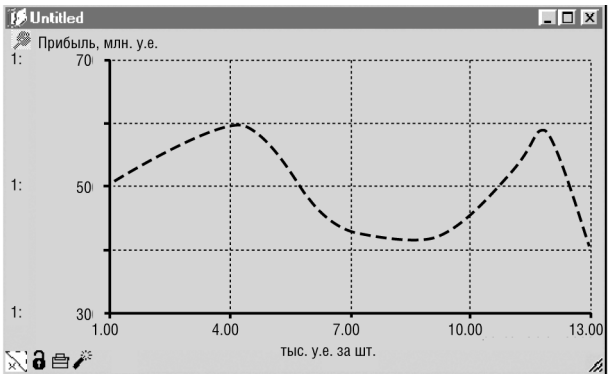
Рассмотрим исходный шаблон модели принятия решений, построенный на принципе шкалы стратегических выборов, размеченных на шкале капитального (ценового) класса выпуска. При помощи приведённой ниже системно-динамической модели проверяется «аспект» производственной стратегии для пар параметров ценового (капитального) класса и избранной цены (*Pb, P*). Модель построена на анализе предпочтений на линейке модельного ряда автомобильной компании, по принципу нарастания капитального класса. Каждое деление означает отдельную производственно-сбытовую стратегию. Ценовой класс изменяется непрерывно в ценовом диапазоне от 2 тыс. до 14 тыс. у.е. Концерн может проверить варианты ценовой и структурной политики в рамках комбинаций «цена–ценовой (капитальный) класс». Очевидно, рассматривалась шкала структурных классов автомобилей более специальная и адаптированная, чем известная классификация автомобилей A, B, C, D, E. В маркетинговых целях и в рамках своих производственных возможностей концерн может назначить цену отличную от базовой метки стратегического класса продукции. Эти обстоятельства обобщают модель, на выходе которой – показатель интегральной эффективности, в данном случае, прибыли от производства и продаж продукции заданной базовой серии.

Стратегия обобщается как

$\text{strat}(Yq_n,\ Pb_n,\ P_n,\ B_n,\ Y_n...)$, при
Yq = 300 тыс.ед.
Pb = 4–10 тыс у.е.
P = 5–13 тыс. у.е.
B = 3–11 тыс. у.е
Yn = 41–63 млн. у.е (прибыль).

В модель введены исходные параметры для всех основных технико-экономических вычислений. Цена и физический объём продаж определяет доход компании капитальный класс – средний бюджет производства продукции. Масштаб производства обеспечивает поправку производственного бюджета на фактор масштаба и т.д. Уровень общих и административных расходов нелинейно зависит от стратегического класса продукции. Изменение этого класса «сдвигает» всю организацию сбытовой системы и соответственно уровень сбытовых и маркетинговых расходов. На этой базе развёртываются численные и концептуальные сценарии операций, деловых программ и последствия кардинальных решений. Модель включается в более широкий контекст по затратам на организацию производства и поддержку всего жизненного цикла продукции структурных категорий. В принципе анализу поддаются все основные характеристики производственно-сбытовых систем. Управление общим сценарием операций концерна осуществляется через параметр капитального класса.

К числу достоинств подхода относится его очевидная применимость и наглядность, что соответствует основным принципам системной динамики как интерактивной доктрины, облегчающие концептуализацию и численную интерпретацию управленческих проблем.

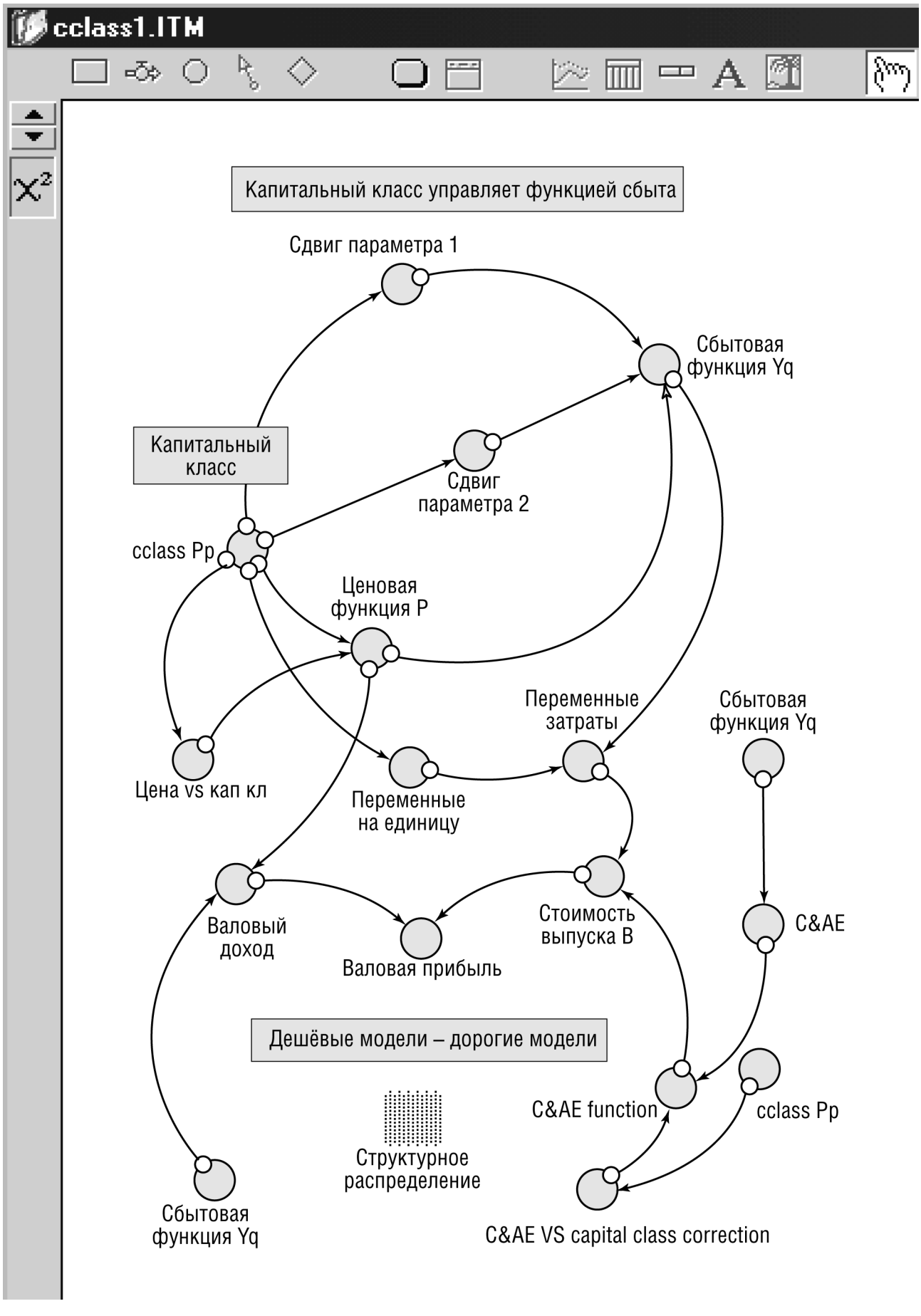


На графике продемонстрирована интегральная эффективность для продукции последовательно нарастающего ценового класса. На нём наблюдается два оптимума, точнее, оптимальной зоны в районе 4–5 и 11–12 тыс. у.е. за единицу. Это можно сопоставить с зоной для «классики» ВАЗ 2105 модельного ряда и зоны модернизированных автомобилей семейства Ваз 2110-«Приора». Эскиз модели по своей логике соответствует анализу продуктовых программ компании А/О «АвтоВАЗ». Он подтверждает вывод о том, что ниша малобюджетных автомобилей и автомобилей эконом-класса остаётся перспективной стратегической нишей российского автопрома, стратегией мирового класса, к которой всё чаше прибегают автомобилестроительные корпорации.

Подобные конструкции приводят к конкретному спектру решений в виде распределений характеристик производственно-сбытовых систем. Они показывают области предпочтительных и наиболее вероятных стратегий, сферу оптимальных компетенции и сравнительных преимуществ концернов и их продукции. В более развитых вариантах модель позволяет выявить весьма тонкие различия в сравнительных условиях производства, стратегическом и операционном статусе, сфере ключевых компетенций концерна. С учётом фактора капитального класса несколько иначе выглядит процесс стратегической конкуренции, когда каждая компания стремится занять наилучшую для себя структурную нишу рынка по характеристикам «капитальный класс – масштаб производства». Происходит размежевание между укрупнёнными ячейками стратегического пространства, где каждое предприятие закрепляет своё место, связанное с её ключевой компетенцией (core competence). В то же время любая ключевая компетенция имеет свою структурную меру и метку на шкале структурного класса.

Параметр усреднённого капитального класса выпуска может вводиться в макроэкономической интерпретации, выполняемой средствами системной динамики. На этой основе возможны новые виды структурной и экономической динамики, а также экономических равновесий, управляемых параметром удельной капиталоёмкости (капитального класса), дополняющего величину рыночной цены. Новые измерения экономического роста и технологического развития могут осуществляться в комбинации с производственными функциями различных классов.

Корпорации подстраивают свои продуктовые линейки к изменяющимся условиям и ориентирам рынка. Например, «сверхмикробюджетные» автомобили за 2,5 тыс. евро или услуги дисконтных перевозчиков в авиационных сообщениях. К подобным маневрам – варьированию стратегическим классом продукции и структурой производственных программ – широко прибегают автомобильные компании «Дженерал Моторс», «Форд» и др. В условиях скачка цен на нефть и сокращения покупательной способности мировые автопроизводители снижают долю выпуска внедорожников и других автомобилей высокого ценового класса. Подстройка стратегии фирмы к изменяющимся условиям конъюнктуры и стратегической конкуренции происходит двумя путями:


cclass1.ITM
Капитальный класс управляет функцией сбыта
Сдвиг параметра 1
Сбытовая функция Yq
Капитальный класс
Сдвиг параметра 2
cclass Pp
Ценовая функция P
Переменные затраты
Сбытовая функция Yq
Цена vs кап кл
Переменные на единицу
Валовый доход
Стоимость выпуска B
C&AE
Валовая прибыль
Дешёвые модели – дорогие модели
C&AE function
cclass Pp
Структурное распределение
Сбытовая функция Yq
C&AE VS capital class correction

1) изменением доли дорогой и дешевой продукции в своём выпуске;

2) разработкой принципиально новых парадигм изделия, построенных на нестандартных комбинациях структурных и ценовых характеристиках изделий.

Подобные революции совершаются и в масштабе отраслей и национальных экономик. Стратегия повышения уровня добавленной стоимости непосредственно связана с повышением доли относительно капиталоёмких изделий при поддержании максимального масштаба их производства.

Исчислению (1–6) поддаются даже такие абстракции, как информационные, сетевые, транспортные и иные потоковые процессы. Любые сетевые структуры, в том числе информационные каналы, сервисные и транспортные системы обычно имеют определённую размерность элементов операционного потока и характеристику пропускной способности (производительности) сетевых узлов и каналов, которые находятся в обратной пропорциональной зависимости. Информационные (сбытовые, транспортные) каналы могут иметь более оптовую (большие партии, малая частота операций) и более розничную характеристику (малые единицы, высокая частота операций). В распределительных системах имеет место изменение этих характеристик от более оптовых к более розничным при движении от центра к периферии сети. Системно-динамическая модель способна в агрегированном и дезагрегированном виде представить базовые соотношения параметров подобных систем и процессов.

Основные выводы

В целях обоснования методов и приёмов моделирования принятия решений средствами системной динамики предложен класс упрощённых алгоритмических конструкций и выражений, обеспечивающих интерпретацию смысла и содержания происходящих процессов.

Предложен принцип капитального класса, позволяющий ввести в модели принятия решений параметр, характеризующий физический облик и структурный статус изделия.

Предложены и обоснованы варианты выражений функции капитального класса, включающих параметр капитального класса при сравнении интегральной эффективности альтернативных производственно-сбытовых стратегий и программ. ■

Литература

1. Райфа Г. Принятие решений. М.: Наука, 1978
2. Diehl,E., Sterman,J.D. (1995) Effects of Feedback Complexity on Dynamic Decision Making. Organizational Behavior and Human Decision Processes, 62, 198–215.
3. Леонов А.И. Управление ассортиментной политикой предприятия». М.: Глобус, 2004.
4. Научно-технический прогресс и капиталистическое воспроизводство.М.: Мысль, 1987.
5. Бусленко Н.П. Моделирование сложных систем М.:, Наука, 1968
6. Productivity Growth and U.S. Competitiveness, Edited by William J.Baumol and Kenneth McLennan., Oxfor University Press, 1985
7. Diehl,E., Sterman,J.D. (1995) Effects of Feedback Complexity on Dynamic Decision Making. Organizational Behavior and Human Decision Processes, 62, 198–215.
8. Mosekilde, E.,Larsen, E.R., Sterman, J.D., Tomsen, J.S.(1992) Nonlinear Mode-Interaction in the Macrjeconomy. Annals of Operation Research, 37, 185–215.
9. Kampmann,C., Haxholdt,C.,Mosekilde E.,Sterman, J.D.(1995) Entrainment in a Dissagregated Economic Long Wave Model. Open Systems and Information Dynamic, 3, 255–274.

РАЗРАБОТКА МЕТОДОЛОГИИ ПРИМЕНЕНИЯ СРЕДСТВА SOLUTION MANAGER В ЗАДАЧЕ ИНТЕГРАЦИИ СИСТЕМЫ SAP

Е.О. Потапова,
студентка магистратуры факультета бизнес-информатики и прикладной математики Нижегородского филиала Государственного университета–Высшей школы экономики
eopotapova@mail.ru

В работе рассматриваются актуальные задачи по организации процесса интеграции сложных корпоративных информационных систем и повышения эффективности данного процесса за счет использования инструментальных средств. В рамках исследования проводится анализ способов применения средств "Solution Manager" при интеграции решений на базе платформы SAP. Автор приводит описание основных возможностей средства Solution Manager и дает рекомендации, как получить от него максимальный эффект в рамках методологии Accelerated SAP (ASAP).

Введение

Ранее нами предложена нейтральная методология реализации или интеграции ERP-систем. Такая методология содержит определение ключевых бизнес-процессов для всех этапов жизненного цикла ERP-системы без привязки к конкретному инструментальному средству. В случае практического применения предложенной методологии необходимо предоставить рекомендации по использованию программных решений определенных производителей. Центральная проблема в данной работе – разработка рекомендаций по совместному использованию предложенной автором методологии и методологии Accelerated SAP (ASAP) для внедрения решений на основе системы SAP. Особое внимание уделяется настройке средства Solution Manager для управления проектами и использования в методологии ASAP.

1. Средство Solution Manager

Внедрение программных решений – непростая задача [3, 4]. Она не должна сказываться на повседневной деятельности компании. Чтобы решить эту проблему, компания SAP предлагает платформу «SAP Solution Manager». «SAP Solution Manager» [1, 2] предоставляющую пользователям бизнес-содержимое и инструментальные средства для эффективной реализации проектов внедрения и глобального развёртывания семейства решений «Управление современным предприятием» (mySAP Business Suite).

Основные функциональные возможности Solution Manager:

1. Управление проектом. Для выполнения задач по управлению проектом есть два приложения:

✧ Project Administration;

✧ Roadmap.

1.1 Приложение Project Administration предназначено для менеджера проекта. С его помощью можно создавать и работать над проектами, управлять системным пространством проекта.

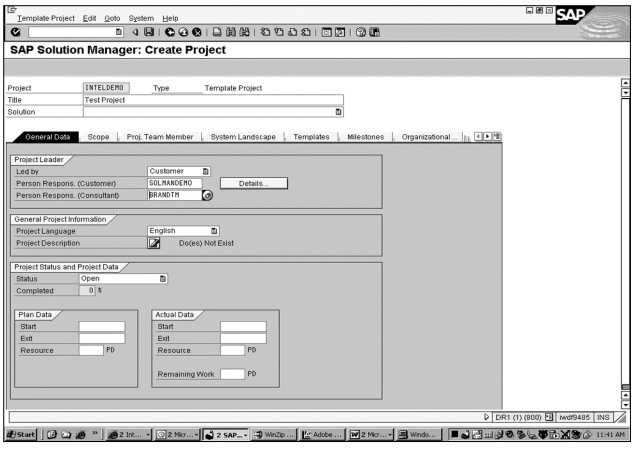


Рис. 1. Интерфейс Solution Manager по созданию нового проекта

С помощью интерфейса данного приложения менеджер проекта может определить:

- ✧ *общую информацию по проекту.* Для этого предназначена закладка General Data. Здесь менеджер может определить ответственных лиц по проекту, язык и статус проекта, прикрепить документ с описанием проекта;
- ✧ *область проекта.* Для этого предназначена закладка Scope. Здесь менеджер может выбрать, какой вариант методологии ASAP будет проводиться в проекте. В зависимости от этого в структуру проекта будет добавлена документация, описывающая тот или иной компонент. Менеджер может указать, для какой индустрии и страны проводится проект. Это позволяет уточнить набор добавляемой документации;
- ✧ *участников проекта.* Для этого предназначена закладка Team Members. Здесь менеджер может сформировать команду проекта, указав привлекаемых работников и их роли в проекте, привлекаемых партнёров и перейти в приложения по администрированию пользователей и ролей;
- ✧ системное пространство проекта – набор логических компонентов; каждый из них отвечает за свой функциональный продукт (например, за CRM или BW). Логический компонент – это логически связанные системы; каждая из них предназначена для различных целей: исследования, проектирование, разработки, тестирование и эксплуатация. При создании логического компонента нужно установить связь между Solution Manager и системами. Это позволяет получать доступ к системам непосредственно через Solution Manager и перемещать конфигурацию из одной среды в другую с помощью автоматизированных средств. Для каждого проекта нужно указывать системное пространство, чтобы проводить конфигурацию системы. Менеджер проекта может сделать это на закладке System Landscape, выбрать из уже определённых логических компонентов необходимые для проекта. На закладке IMG Projects для систем, указанных в системном пространстве, менеджер может сгенерировать конфигурационные проекты IMG, с помощью которых проводится вся конфигурация в SAP компонентах;
- ✧ *ключевые моменты.* Для этого предназначена закладка Milestones. Здесь менеджер может указать ключевые моменты, которые должны быть получены в ходе проекта;

* *стандарты проекта.* С помощью этого экрана менеджер может указать статусы, ключевые слова и типы документов, используемые в проекте.

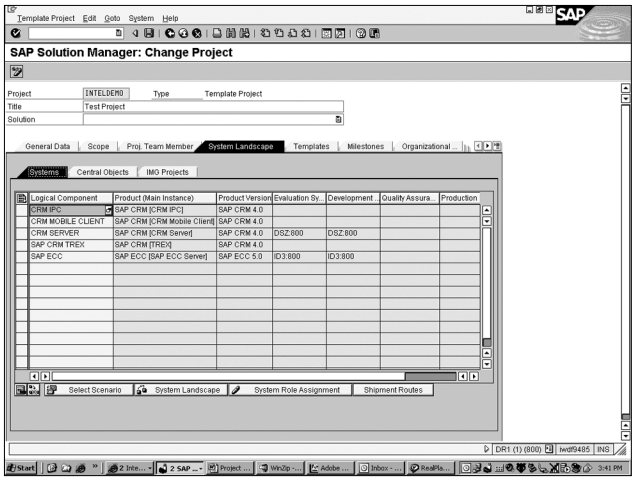


Рис. 2. Интерфейс Solution Manager по определению системного окружения

1.2 Приложение Roadmap предназначено для управления проектом. В нём для выбранного проекта отражается содержание методологии, согласно которой этот проект должен проводиться и которая была привязана к нему в приложении Project Administration на закладке Scope. При входе в это приложение пользователь видит графическое приложение методологии, которая присоединена к проекту. В графическом представлении показывается пять фаз любой методологии и ссылки на пакеты работ. Если кликнуть на какую-либо ссылку, пользователь перейдёт к структурному представлению методологии и увидит содержание выбранного им пакета работ.

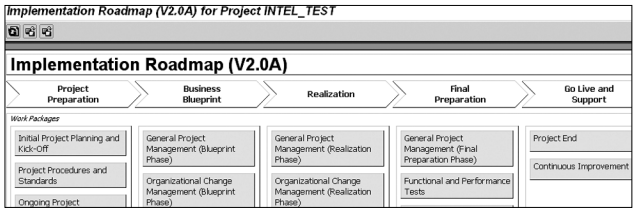


Рис. 3. Графическое представление методологии

Настройки системы позволяют избежать графической формы, и уже при входе видеть структурное представление методологии.

Структурное представление изображено на *рис. 4.*

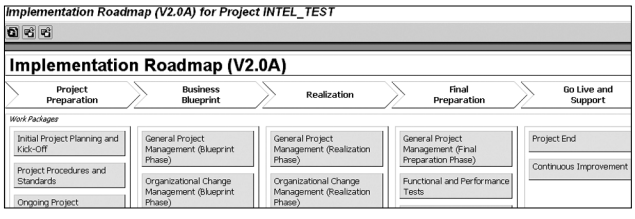


Рис. 4. Структурное представление методологии

В этом представлении можно выделить 3 области:

- *структура методологии* (слева). Здесь находится иерархия, описывающая выбранную методологию. Она содержит: фазы (их всегда 5), группы целей, цели, результаты, действия и задачи. Фазы состоят из групп целей, группы целей – из целей, а цели содержат результаты, которые нужно получить, чтобы цель была достигнута, и действия для достижения цели и получения результатов. Действия могут быть разбиты на более мелкие задачи;
- *описание* (справа сверху). Для каждого узла в структурной иерархии существует достаточно подробное описание. Оно содержит информацию, помогающая пользователю выполнить требования, достичь цели или получить результат. Для различных элементов описание различно. Для каждого узла указываются следующие элементы:
  - цель (фазы, действия и т.д.);
  - предварительные требования (что должно быть сделано);
  - основной результат;
  - процедура;
- дополнительная информация (справа снизу). Для каждого узла в структурной иерархии возможно указать набор дополнительной информации. Для этого предназначен набор закладок внизу страницы.

На первой закладке Accelerator представлен набор документов, поставляемый SAP. Среди них содержатся шаблоны, образцы, веб ссылки и другая информация, полезная для выполнения работ и получения заданных результатов.

На второй закладке Status/Notices пользователь может заполнить новую запись/комментарий к определённому шагу в методологии или изменить статус этапа. Есть четыре варианта статусов: Open, Completed, In process, Error. При установке статуса в структуре появляется знак у данного шага. Статус будет виден в графическом представлении методологии.

На третьей закладке Project Team Members указываются имена участников, ответственных за выполнение данного этапа, шага или получение определённого результата.

На четвертой закладке Messages пользователь может создавать и просматривать сообщения по определённому шагу в методологии. Для каждого сообщения можно определить основную тему, область, приоритет и исполнителя.

На пятой закладке Project Documentation пользователь может загружать различные документы.

На шестой закладке Keywords пользователь может определять ключевые слова для поиска документов.

В данном приложении структуру методологии можно выгрузить в формат HTML или в виде шаблона для системы MS Project.

2. Создание концептуального проекта. Для создания концептуального проекта предназначено приложение *Business Blueprint.*

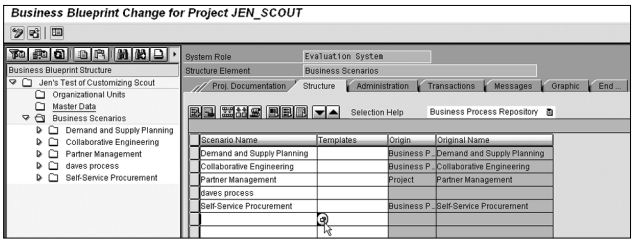


Рис. 5. Интерфейс приложения Business Blueprint

В нём пользователю необходимо создать иерархическую структуру бизнес-сценариев и бизнес-процессов, которые будут затрагиваться проектом. Формирование этой структуры происходит с помощью закладки Structure, где отображается содержимое выбранной слева папки. Для формирования иерархии Solution Manager предоставляет специальный репозиторий, где хранятся сценарии и процессы, согласно которым функционируют различные SAP системы.

Для каждого сценария, процесса или узла можно добавлять документы на закладках General Documentation и Project Documentation. Закладка General Documentation содержит документацию, добавленную изначально к сценарию или имевшуюся в шаблонном проекте, а закладка Project Documentation может использоваться для хранения общей информации, касающейся того или иного узла.

По каждому элементу в структуре можно указать статус работы, ответственных лиц и другую административную информацию на закладке Administration. Для различных узлов можно указать транзакции SAP систем, имеющие отношения к данной функциональности, создать сообщения к службе поддержки или запросы на изменения, которые будут отправлены в единый репозиторий для обработки. Для сценариев и бизнес-процессов дополнительно можно просмотреть графическое представление элементов на закладке Graphics.

На основе определённой структуры Solution Manager может сгенерировать документ в формате .doc, содержащую требуемую информацию. При генерации этого документа пользователь может

выбрать документацию, для вставки в документ, необходимые графические представления, задать условие для структурных элементов и условия отображения.

3. Конфигурация. Для описания конфигурации в Solution Manager предназначено приложение Configuration. В нём отображается структура выбранных ранее бизнес-сценариев и бизнес-процессов. Для каждого элемента в структуре активны следующие закладки:

- ✧ General Documentation;
- ✧ Project Documentation;
- ✧ Structure;
- ✧ Administration;
- ✧ Transactions;
- ✧ Configuration;
- ✧ Development;
- ✧ Test Cases;
- ✧ Messages;
- ✧ Graphic;
- ✧ Training Material;
- ✧ End User Roles.

Большинство из них соответствуют закладкам приложения Business Blueprint и выполняют аналогичные функции. Рассмотрим назначение закладок Configuration, Development и Test Cases.

Закладка Configuration предназначена для описания конфигурации, сделанной в отношении некоторого шага. Здесь можно прикреплять различные объекты, содержащие конфигурацию, либо описывающие её. В этом приложении конфигурационные объекты могут быть подгружены только из системы, выполняющей роль по разработке для выбранного логического компонента, поэтому важно иметь предопределённое системное пространство.

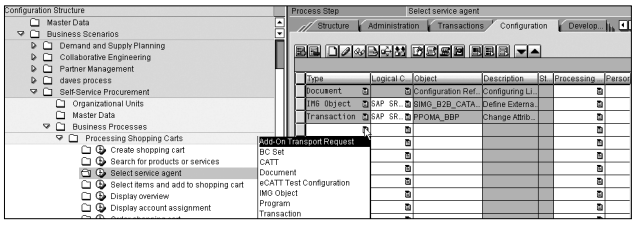


Рис. 6. Приложение по конфигурации. Определение конфигурационных объектов

Закладка Development имеет схожее назначение, но на ней фиксируются объекты, с помощью которых проводится изменение и расширение стандартной функциональности.

Отдельную роль играет *закладка Test Cases.* На ней для различных элементов можно загружать описание вариантов тестирования, впоследствии включаемые в план тестирования.

4. Тестирование. Для организации тестирования в Solution Manager предназначены приложения Test Plan Management и Test Execution.

4.1 Приложение Test Plan Management предназначено для управления планом тестирования. В нём менеджер по тестированию может создать план тестирования для определённого проекта и для различных уровней тестирования (интеграционного, производительности, сдачи-приёмки и т.д.). Основное содержимое проекта будут составлять варианты тестирования, добавленные к структуре концептуального проекта в приложении по конфигурации. При создании нового тест-плана пользователю необходимо выбрать набор тестовых вариантов, предлагаемых системой, важных для данного уровня тестирования. Для каждого тест-плана менеджер может создать набор тестовых пакетов. Тестовый пакет представляет собой набор тестовых вариантов, сгруппированных по определённому принципу. Тестовый пакет – это единица работы тестера, и к каждому пакету должен быть добавлен исполнитель работ (или несколько исполнителей).

Система собирает статистику по исполнению тестовых вариантов и показывает её в приложении Test Plan Management.

4.2. Test Execution. Данное приложение – основное для тестера. В нём пользователь видит все тестовые пакеты из различных тестовых планов, для которых пользователь является исполнителем. Здесь тестер может запустить на выполнение тесты и зафиксировать полученный результат. Если в ходе тестирования обнаружена ошибка, тестер может создать сообщение на поддержку, которое будет обрабатываться, согласно установленному процессу.

5. Управление системными компонентами. Системные компоненты – одна из наиболее важных областей управления проектом. Для каждого проекта необходимо иметь 5 систем:

- ✧ исследования;
- ✧ разработки;
- ✧ тестирования;
- ✧ производственная;
- ✧ обучение.

Для различных продуктов – CRM, ECC или BW – должны быть в наличии все перечисленные системы с одинаковыми версиями продуктов. Необходима связь между данными системами, так как для повышения качества и избегания ошибок перенос

конфигурации должен быть автоматическим. Поддержка систем и обеспечение связи между ними – очень сложная задача; для её решения Solution Manager предлагает приложение System Landscape Management.

В этом приложении возможно:

- ✧ хранить данные о серверах, базах данных и системах;
- ✧ определять логические компоненты, представляющие собой системы и роли, для которых системы предназначены;
- ✧ определять продукты, представляющие собой различные по функциональности системы;
- ✧ создавать связи между системами в одном компоненте.

Основными пользователями данного приложения – системные администраторы, отвечающие за техническое обеспечение.

6. Создание учебных материалов. Обучение пользователей – чрезвычайно важный процесс для успешного внедрения ERP-системы. Чтобы наилучшим образом подготовиться к обучению и провести его, в Solution Manager существует приложение Learning Maps. В нём для заданного проекта можно разработать структуру учебных курсов, учебные материалы и приложения.

7. Управление изменениями. Процедура управления изменениями чрезвычайна важна для успешного завершения проектов. В Solution Manager эта процедура основана на работе с запросами на изменения. Для внесения какого-либо изменения пользователю предлагается создать запрос на изменения. Для этого предназначено приложение Create Change Requests. Создание запроса может происходить либо через запрос на поддержку, либо через сообщение об ошибке. Далее запрос на изменение должен быть подтверждён. Если это происходит, из запроса на изменение создаётся коррекция. Для каждой коррекции должен быть добавлен специальный лист задач, в котором указываются необходимые шаги, чтобы решить проблему, а изменения перевести в производственную систему.

2. Рекомендации по использованию средства Solution Manager для управления проектами по внедрению систем SAP

2.1 Пример использования средства Solution Manager для управления проектами. Для более глубокого изучения средства Solution Manager на основе изученной методологии ASAP и с учётом собственного разработанного унифицированного процесса внедрения сложных систем проведён небольшой проект. Он не выводился в производственную среду и в нёго не добавлялись реальные данные.

В ходе этого проекта сделаны следующие шаги:

1. На фазе Project Preparation создан проект. Для него указана общая информация по проекту, а в качестве основной методологии – указана методология ASAP. (*см. рис. 7, 8*).

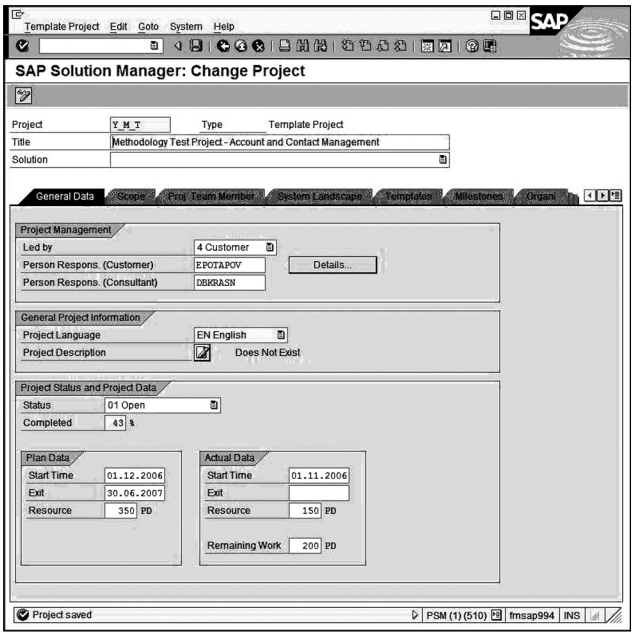


Рис. 7. Создание нового проекта. Указание общей информации

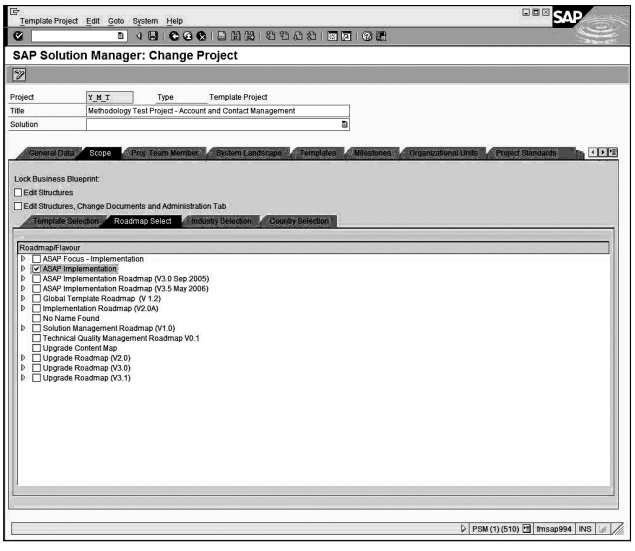


Рис. 8. Создание нового проекта. Выбор методологии

Чтобы создать требуемых участников команды, использована транзакция по администрированию

пользователей, где созданы новые пользователи для участников команды и приписаны им необходимые роли. Участники команды приписаны к проекту в транзакции по Project Administration на закладке Project Team Members (*см. рис. 9*).

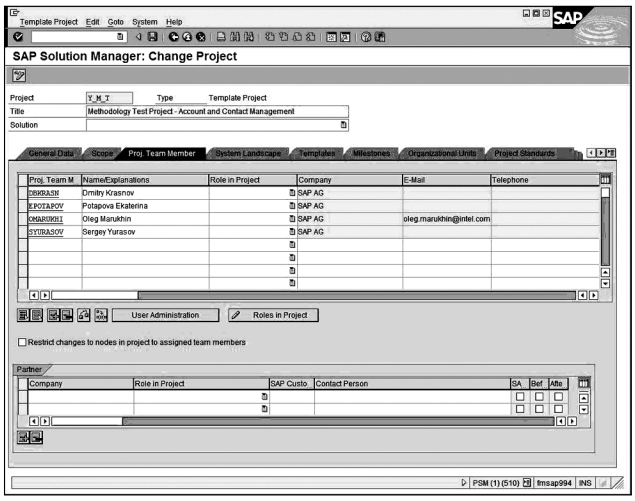


Рис. 9. Создание нового проекта. Выбор участников

Далее на закладке System Landscape добавлены необходимые логические компоненты для проекта. Частично они выбраны из имеющихся, а частично созданы в транзакции System Landscape (*см. рис. 10, 11*).

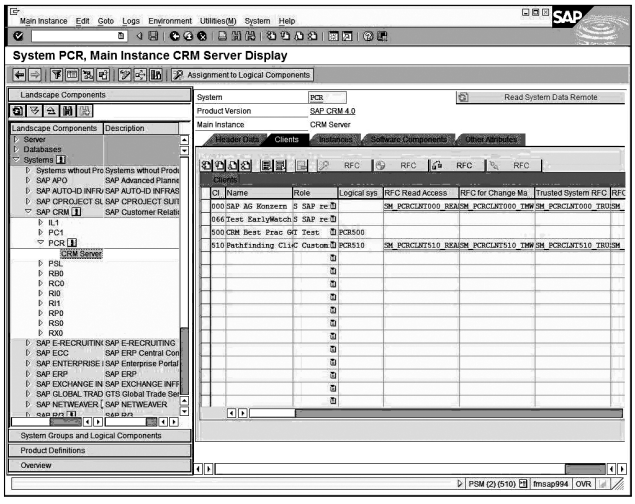


Рис. 10. Создание системных компонентов

А на закладке Project Standards определены ключевые слова, статусы и типы документов для использования в проекте (*см. рис. 12*).

В транзакции Roadmap методология выгружена в формате MS Project. (*см. рис. 13*).

На основе полученного документа составлен план работ и план использования ресурсов. Полученные документы прикреплены к основному узлу иерархии (*см. рис. 14*).

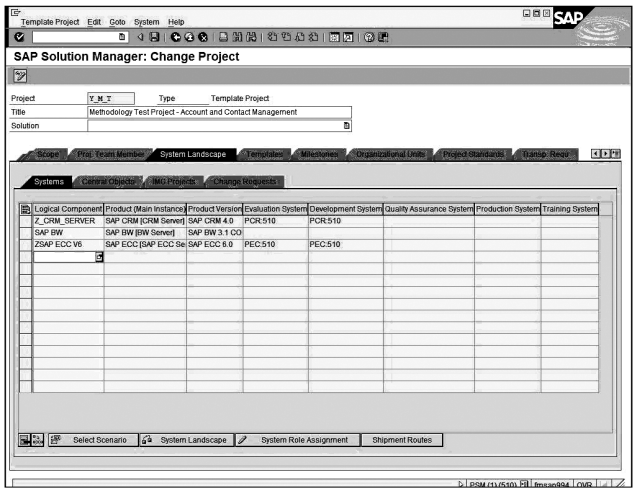


Рис. 11. Определение системного пространства

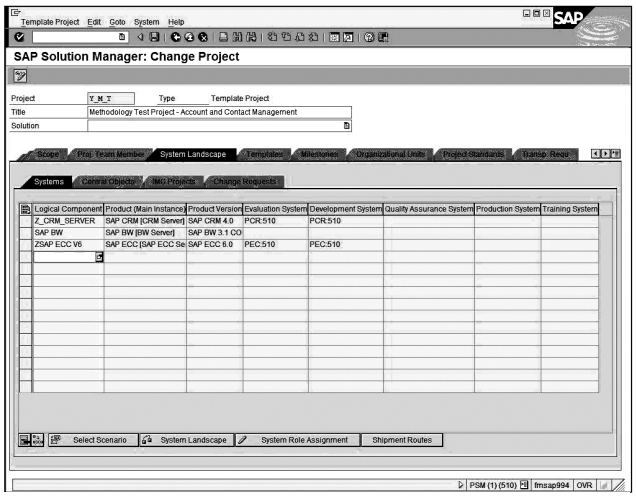


Рис. 12. Определение ключевых слов, статусов и документов

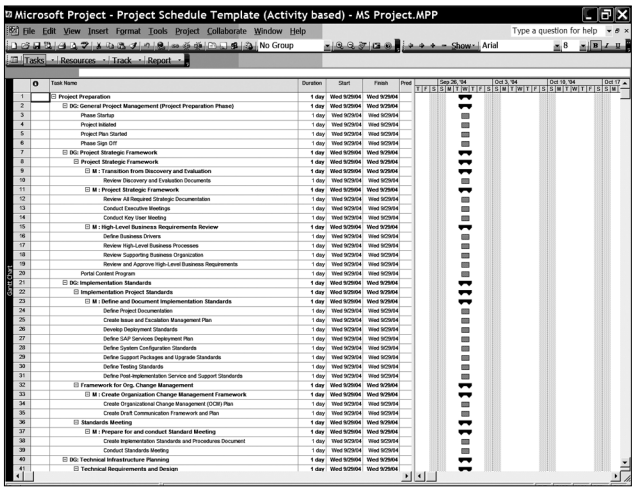


Рис. 13. План проекта в формате MS Project

На основе планов за соответствующие результаты и действия назначены исполнители (*см. рис. 15*).

Дополнительно требующиеся документы добавлены к структуре Roadmap

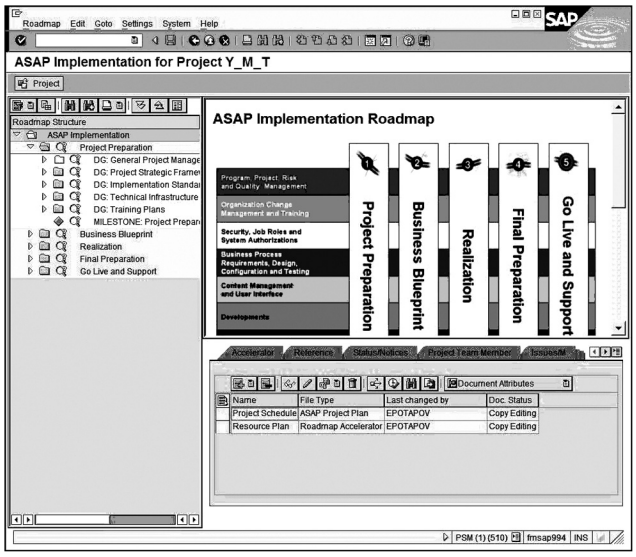


Рис. 14. Документы, прикреплённые к иерархии

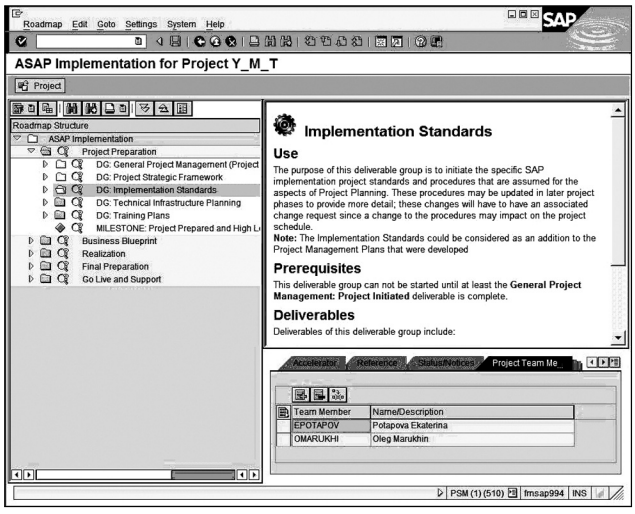


Рис. 15. Назначение исполнителей на основе планов за соответствующие результаты и действия

В транзакции Learning Map создан новый пакет по обучению и задана его ориентировочная структура (см. рис. 16).

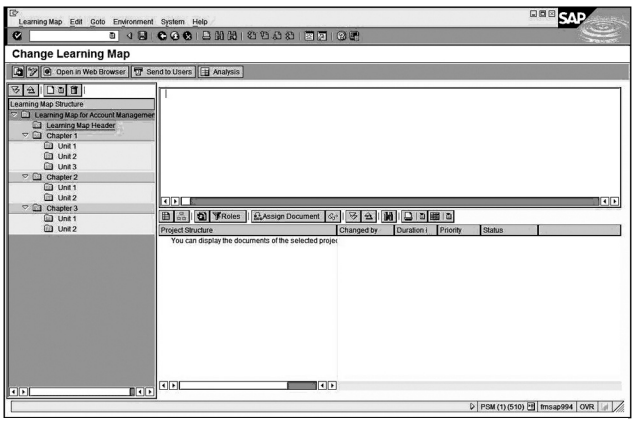


Рис. 16. Структура пакета обучения

2. На фазе Business Blueprint составлена структура концептуального проекта; добавлены основные сценарии, процессы, транзакции и документы, объясняющие цели и функциональность, затрагиваемую на разных шагах бизнес-процессов (*см. рис.17*).

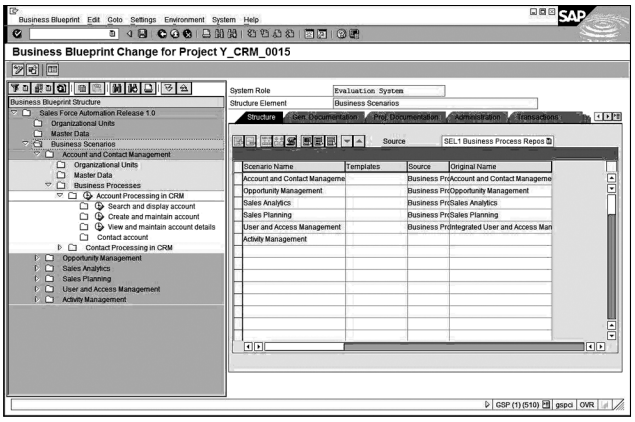


Рис. 17. Структура процессов концептуального проекта

На основе проекта сделан анализ несоответствий, в котором указаны основные несовпадения требований клиентов и возможностей SAP (*см. рис. 18*).

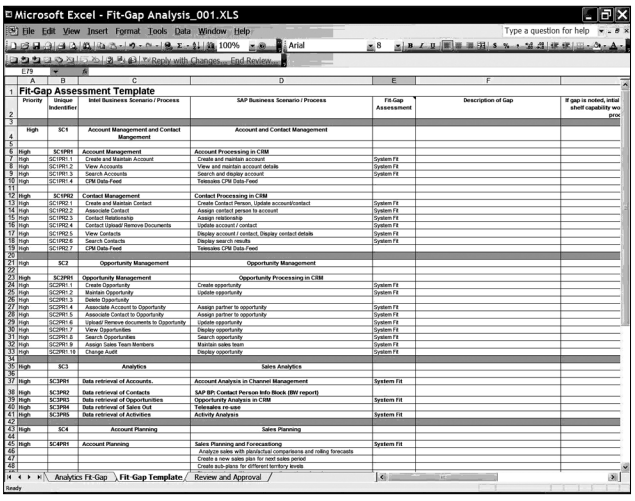


Рис. 18. Анализ несоответствий

На основе анализа несоответствий внесены изменения в структуры конфигурационного проекта и сформулированы требования по конфигурации и модификации. На основе получившейся структуры сгенерирован Business Blueprint документ, проведена проверка систем для разработки в транзакции System Landscape и настроены каналы связи с ними.

Для каждой из этих систем в транзакции Project Administration на закладке Project IMG сгенерированы конфигурационные проекты. Для них из полного списка конфигурационных объектов, предоставляемых SAP, выбраны те, которые могут потребоваться при конфигурации (*см. рис. 19*).

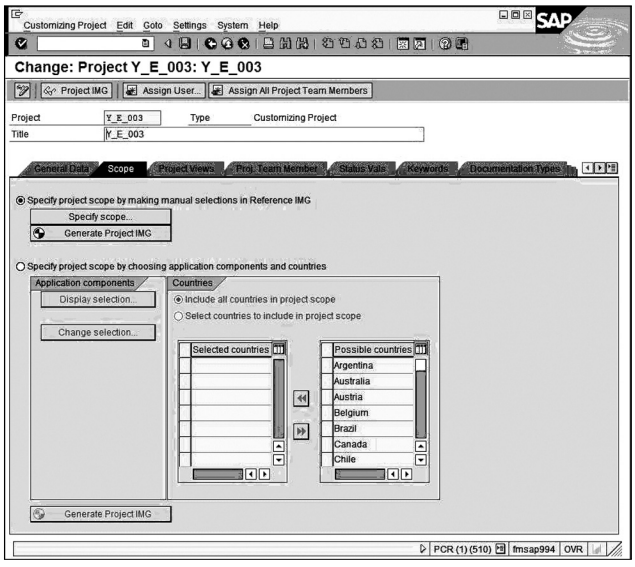


Рис. 19. Определение области Project IMG

Дополнительно определены шаблоны материалов для обучения и добавлены в список возможных документов в проекте.

Для полученной структуры на закладке Configuration перечислены различные конфигурационные объекты. Их необходимо затронуть для выполнения конфигурации (*см. рис. 20*).

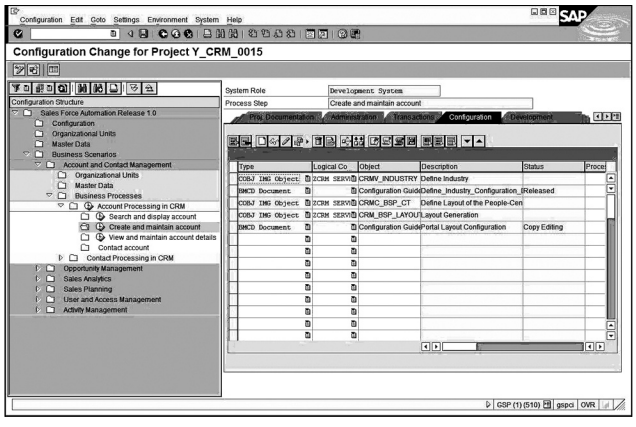


Рис. 20. Определение конфигурационных объектов и проектирование конфигурации

В результате для проекта сгенерирован Configuration Guide. В нём перечислена требуемая конфигурация и методы её осуществления.

3. На фазе Realization исполнена вся требуемая конфигурация с помощью объектов из Project IMG, добавленных к структуре в предыдущей стадии.

К элементам, к которым применялась конфигурация, и к бизнес-процессам на закладке Test Cases добавлены варианты тестов (*см. рис. 21*).

Для осуществления тестирования проведена проверка системы тестирования и настроен доступ

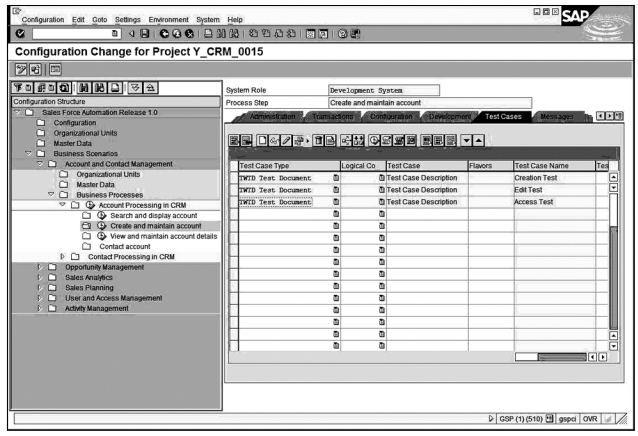


Рис. 21. Определение тестовых вариантов

к ней. В транзакции Test Plan Management создан тест-план (*см. рис. 22*).

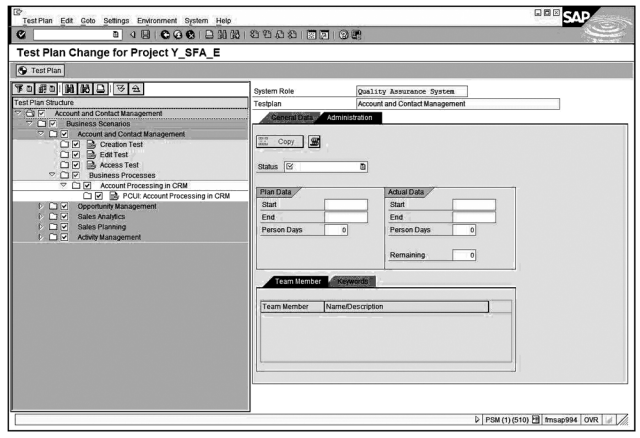


Рис. 22. Создание тестового плана

В нём все тестовые варианты разбиты на пакеты, и для каждого пакета добавлен тестер (*см. рис. 23*).

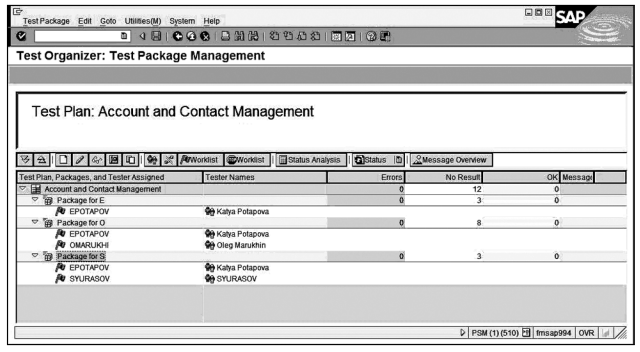


Рис. 23. Создание тестовых пакетов для различных исполнителей

Для выполнения тестирования использовалась транзакция Test Execution, где для каждого тестового вариант отмечен статус выполнения тестов и создан отчёт о результате проведения теста (*см. рис. 24, 25)*.

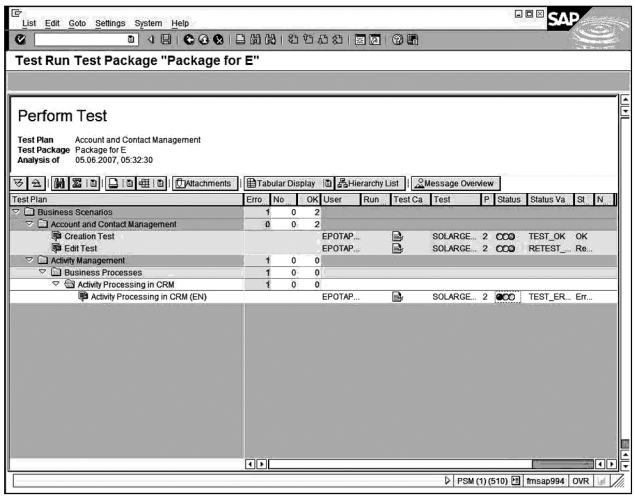


Рис. 24. Интерфейс для исполнения тестов

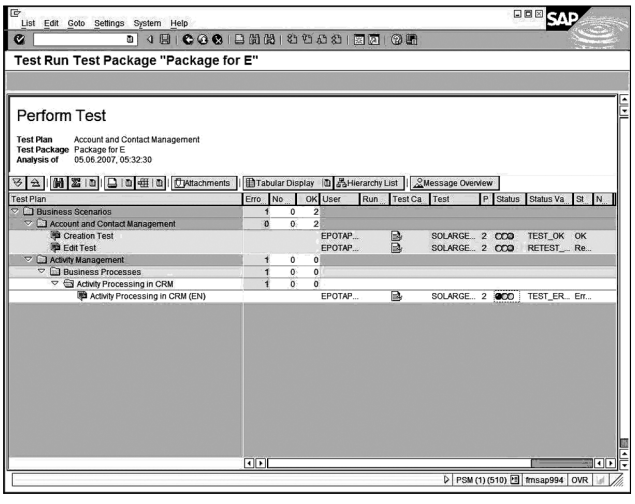


Рис. 25. Описание результатов теста

Для каждой возникающей ошибки поднято сообщение по поддержке. В зависимости от результатов анализа оно либо отклонено, либо для него создан запрос на изменение и коррекция.

5.2 Общие рекомендации по использованию средства Solution Manager для управления проектами

Результаты проведенной работы и знания, полученные в её ходе, позволили составить следующую методологию по использованию средства Solution Manager для проведения проектов согласно технологии ASAP(*табл. 1*).

Таблица 1

№№ п/п	Фазы/ Этапы	Рекомендуемые действия
1	**Подготовка проекта**	
1.1	Управление проектом	В транзакции Project Administration: ✦ создать проект; ✦ создать общую информацию по проекту на закладке General Data; ✦ собрать методологию разработки на закладке Scope -> Roadmap Select; ✦ выбрать участников для работы в проекте на закладке Project Team Members. В транзакции Roadmap: ✦ выгрузить Roadmap как документ для MS Project; ✦ преобразовать полученный документ в «Resource Plan and Schedule» и «Project Schedule»; ✦ прикрепить полученные документы к корневому узлу структуры Roadmap; ✦ исходя из «Resource Plan and Schedule» прикрепить участников, ответственных за выполнение задач или получения результатов, к соответствующим узлам структуры Roadmap; ✦ на основе имеющихся типов документов и прикреплённых к ним шаблонов создать другие требуемые документы и прикрепить их к соответствующим узлам структуры Roadmap; ✦ использовать закладки Status/Notices, чтобы формально отмечать завершение и корректность целей.
1.2	Разработка стратегической основы проекта	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать «Project Description», в который должны войти все требуемые результаты этой цели, и прикрепить полученный документ к соответствующему узлу структуры Roadmap. ✦ Прикрепить полученный документ как Project Description в транзакции Project Administration.
1.3	Разработка стандартов внедрения	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые стандарты. Прикрепить их к соответствующему узлу структуры Roadmap. ✦ В транзакции Project Administration, на закладке Project Standards -> Documentation Types согласно определённым стандартам создать или выбрать те виды документов, которые будут использоваться, и прикрепить к ним соответствующие шаблоны.

см. продолжение

Продолжение таблицы 1

№№ п/п	Фазы/ Этапы	Рекомендуемые действия
1	**Подготовка проекта (продолжение)**	
1.4	Планирование технической инфраструктуры	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать «Solution Landscape Deployment Plan» и прикрепить его к соответствующему узлу структуры Roadmap. ✦ В транзакции SMSY выбрать наиболее подходящие компоненты для проекта, или создать собственные. ✦ В транзакции Project Administration добавить выбранные логические компоненты к проекту на закладке System Landscape -> Systems. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать остальные требуемые документы (см. ASAP) и прикрепить их к соответствующим узлам структуры Roadmap. ✦ В транзакции SU01 прикрепить роли для участников команды.
1.5	Планирование обучения	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы (см. ASAP) и прикрепить их к соответствующим узлам структуры Roadmap. ✦ В транзакции SOLAR_LEARNING_MAP создать прототип курса с разделением на части и/или темы. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать «End User Training and Documentation: Needs Assessment», «End User Training and Documentation Work Plan», «End User Proposal» и прикрепить к соответствующим узлам структуры Roadmap.
2	**Создание концептуального проекта**	
2.1	Управление проектом	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикрепить их к соответствующим узлам структуры Roadmap. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать Request for Change и прикрепить к соответствующим узлам. ✦ Использовать Change Request Management для обработки Requests for Changes. ✦ Вести Issue Log в транзакции Roadmap на закладке Issue/Messages. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов вести единый документ для учёта рисков (Risk and Response Log). ✦ Отмечать принятие каждой цели в фазе как комментарий и изменение статуса на закладке Status/Notices. ✦ Найти документ «Communication Matrix», внести в него изменения, прикрепить к соответствующему узлу структуры Roadmap и прикрепить ссылки на него. ✦ Отметить статусы объектов в Roadmap и создать комментарии к ним
2.2	Обновление плана обучения	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать документ «End User Training Curriculum» и прикрепить его к соответствующему узлу структуры Roadmap. ✦ В транзакции SMSY выбрать систему для проведения обучения или убедится, что подходящей системы нет – сформировать требования. ✦ В транзакции SOLAR_LEARNING_MAP пересмотреть структуру обучения. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы (см. ASAP) и прикрепить их к соответствующим узлам структуры Roadmap. ✦ Разработать шаблоны учебных материалов как новые типы документов или добавить шаблоны к уже существующим типам документов, предназначенных для обучения.
2.3	Определение бизнес-требований к предоставляемому решению	В рамках данной цели в SM можно: ✦ в транзакции Business Blueprint выбрать из уже имеющихся и присоединить к проекту основные элементы организационной структуры, элементы данных и бизнес-сценарии; ✦ создать собственные сценарии и процессы; ✦ в транзакции Business Blueprint добавить необходимые описания и требования к элементам на закладке Project Documentation; ✦ транзакции Business Blueprint добавить необходимые транзакции для проверки и анализа элементов.
2.4	Управление изменениями	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требуемые документы и прикреплять их к соответствующим узлам в структуре Roadmap. ✦ Определить роли конечных пользователей в транзакции Business Blueprint.
2.5	Разработка детальных бизнес-требований	✦ Из транзакции Business Blueprint запустить все транзакции для различных элементов и провести анализ возможных путей достижения цели. ✦ Провести реинжениринг бизнес-процессов, изменив их графическую структуру, получить модель To-Be. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требования к окончательным бизнес-процессам и прикреплять их к соответствующим узлам в структуре Business Blueprint. ✦ Автоматически сгенерировать Business Blueprint Document.

см. продолжение

Продолжение таблицы 1

№№ п/п	Фазы/ Этапы	Рекомендуемые действия
2	**Создание концептуального проекта (продолжение)**	
2.6	Определение процедур разработки и концептуальное проектирование разработок	✦ В транзакции SMSY проверить (создать) компоненты для development. ✦ В транзакции Project Administration создать IMG-проекты для всех development компонентов. ✦ Определить области IMG-проектов для систем. ✦ В транзакции Configuration добавить конфигурационные объекты, которые могут потребоваться. ✦ В транзакции Configuration добавить описание модификаций, которые могут потребоваться. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикреплять их к соответствующим узлам в структуре Roadmap. ✦ На базе полученных стандартов внести изменения на закладке Project Standards, в транзакции Project Administration. ✦ В транзакции SU01 создать пользователей, которые будут работать на фазе Development
2.7	Определение требований безопасности	✦ Создать прототипы ролей для использования. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикреплять их к соответствующим узлам в структуре Roadmap.
2.8	Проектирование технической архитектуры и инсталляция проекта	✦ После установки дополнительного оборудования в транзакции SMSY создать дополнительные системы. ✦ Настроить систему доступа к новым системам. ✦ Установить в логический компонент проекта Development system. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикреплять их к соответствующим узлам в структуре Roadmap. ✦ В транзакции Project Administration на закладке System Landscape-> IMG Projects генерировать Project IMG для конфигурации.
3	**Реализация**	
3.1	Управление проектом	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать Request for Change и прикрепить к соответствующим узлам. ✦ Использовать Change Request Management для обработки Requests for Changes. ✦ Вести Issue Log в транзакции Roadmap на закладке Issue/Messages. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов вести единый документ для учёта рисков (Risk and Response Log). ✦ Отмечать принятие каждой цели в фазе как комментарий и изменение статуса на закладке Status/Notices. ✦ Найти документ «Communication Matrix», внести в него изменения и прикрепить к соответствующему узлу структуры Roadmap; прикрепить ссылки на него. ✦ Отметить статусы объектов в Roadmap и создавать комментарии к ним.
3.2	Управление изменениями	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap.
3.3	Обучение пользователей и подготовка документации	✦ Разработать шаблоны учебных материалов как новые типы документов или добавить шаблоны к уже существующим типам документов, предназначенных для обучения. ✦ В транзакции SOLAR_LEARNING_MAP определить окончательную структуру курса, составить точное описание всех разделов. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикрепить их к соответствующим узлам структуры Roadmap. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать документы для обучения и прикрепить их к соответствующим узлам в структуре Configuration на закладке Training Materials. ✦ В транзакции SOLAR_LEARNING_MAP распределить учебные материалы по соответствующим разделам. ✦ В транзакции SMSY определить Training System для логического компонента проекта, настроить доступ (RFC-connection, user Ids). ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикрепить их к соответствующим узлам в структуры Roadmap или Configuration. ✦ В транзакции Configuration изменить System Role и проделать конфигурацию или тесты внутри системы для обучения.
3.4	Определение конфигураций, интерфейсов, интеграционных пакетов	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать необходимые документы и прикрепить их к соответствующим узлам в структуре Configuration на закладке Project Documentation. ✦ Провести требуемую конфигурацию через объекты, прикреплённые на закладке Configuration. ✦ Определить метод доработки элементов, добавив описание всех изменений на закладке Development. ✦ Создать тестовые ситуации для тестирования форм и добавить их на закладку Test Cases.

см. продолжение

Продолжение таблицы 1

№№ п/п	Фазы/ Этапы	Рекомендуемые действия
3	**Реализация (продолжение)**	
3.5	Основная конфигурация	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать необходимые документы и прикрепить их к соответствующим узлам в структуре Configuration на закладке Project Documentation. ✦ Описать требуемую конфигурацию в виде различных элементов и документов на закладке Configuration. ✦ Использовать сгенерированный IMG для проведения конфигурации. ✦ Определить тестовые варианты для различных элементов на закладке Test Cases. ✦ В транзакции Test Plan Management сгенерировать тест-план, создать в нём тест-пакеты, назначить исполнителей, выполнить тесты и записать отчёты о них.
3.6	Обеспечение качества	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать необходимые документы и прикрепить их к соответствующим узлам в структуре Roadmap. ✦ В транзакции SMSY создать систему для роли Quality Assurance, настроить к ней доступ из Solution Manager. ✦ В транзакции SMSY определить системы на роль Quality Assurance для логических компонентов проекта.
3.7	Планирование сдачи проекта и поддержки	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать необходимые документы и прикрепить их к соответствующим узлам в структуре Roadmap.
3.8	Внедрение авторизаций и систем безопасности	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать необходимые документы и прикрепить их к соответствующим узлам в структуре Roadmap. ✦ Создать требуемые роли и авторизации для ограничения доступа. ✦ Создать тестовые случаи для тестирования ролей и авторизаций. ✦ Изменить тест-план. ✦ Собрать отчёты по тестам.
3.9	Завершение конфигурации	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать необходимые документы и прикрепить их к соответствующим узлам в структуре Configuration. ✦ Описать сделанные доработки с помощью объектов(программ, документов, транзакций), прикрепив их к соответствующим узлам в структуре Configuration на закладке Development. ✦ Описать сделанную конфигурацию с помощью объектов (программ, документов, транзакций), прикрепив их к соответствующим узлам на закладке Configuration. ✦ Определить завершение конфигурации по статусам всех задач, документов и объектов. ✦ Определить случаи для тестирования. ✦ В транзакции Test Plan Management сгенерировать тест-план, создать в нём тест-пакеты, назначить исполнителей, выполнить тесты и записать отчёты о них.
3.10	Финальное интеграционное тестирование	✦ В транзакции Configuration определить случаи для тестирования. ✦ В транзакции Test Plan Management создать Final Integration Test Plan.
3.11	Инсталляция производственной среды	✦ В транзакции SMSY создать систему для роли Production, настроить к ней доступ из Solution Manager. ✦ В транзакции SMSY определить системы на роль Production для логических компонентов проекта.
3.12	Планирование тестирования производительности	✦ В транзакции Configuration создать случаи для тестирования. ✦ В транзакции Test Plan Management создать требуемые тестовые планы, тест-пакеты, назначить исполнителей, собрать результаты тестов.
4	**Завершающая подготовка**	
4.1	Управление проектом	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать Request for Change и прикрепить к соответствующим узлам. ✦ Использовать Change Request Management для обработки Requests for Changes. ✦ Вести Issue Log в транзакции Roadmap на закладке Issue/Messages. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов вести единый документ для учёта рисков (Risk and Response Log). ✦ Отмечать принятие каждой цели в фазе как комментарий и изменение статуса на закладке Status/Notices. ✦ Найти документ «Communication Matrix», внести в него изменения, прикрепить к соответствующему узлу структуры Roadmap и прикрепить ссылки на него. ✦ Отметить статусы объектов в Roadmap и создать комментарии к ним.

см. окончание

Окончание таблицы 1

№№ п/п	Фазы/ Этапы	Рекомендуемые действия
5	**Использование и поддержка**	
4	**Завершающая подготовка (продолжение)**	
4.2	Завершающая подготовка заказчика	✦ Выгрузить окончательную версию программы обучения и разослать её пользователям.
4.3	Завершающая подготовка производственной среды и системы поддержки	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap. ✦ Использовать Service Desk для обеспечения поддержки пользователей.
4.4	Сдача производственной системы	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap.
5	**Использование и поддержка**	
5.1	Управление проектом	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создать Request for Change и прикрепить к соответствующим узлам. ✦ Использовать Change Request Management для обработки Requests for Changes. ✦ Вести Issue Log в транзакции Roadmap на закладке Issue/Messages. ✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов вести единый документ для учёта рисков (Risk and Response Log). ✦ Отмечать принятие каждой цели в фазе как комментарий и изменение статуса на закладке Status/Notices. ✦ Найти документ «Communication Matrix», внести в него изменения, прикрепить к соответствующему узлу структуры Roadmap и прикрепить ссылки на него. ✦ Отметить статусы объектов в Roadmap и создать комментарии к ним.
5.2	Обеспечение первоначальной поддержки	✦ Использовать транзакцию Solution Operation для контроля за процессами и транзакциями. ✦ Использовать транзакцию SOLAR_LEARNING_MAP для усовершенствования учебных материалов и распространения последних версий. ✦ Использовать транзакцию SOLAR_LEARNING_MAP для создания учебных материалов для последующего обучения.
5.3	Завершение сдачи системы	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap.
5.4	Закрытие проекта	✦ На основе имеющихся типов документов и прикреплённых к ним шаблонов создавать требуемые документы и прикреплять их к соответствующим узлам структуры Roadmap.

Литература

1. Помощь по средству Solution Manager http://help.sap.com/saphelp_sm310/helpdata/en/index.htm.
2. Описание методологии ASAP http://service.sap.com/education/asap/index.com.
3. Linda K. Lau, «Managing Business with SAP: Planning, Implementation, and Evaluation».
4. Вивек Кале, «Внедрение SAP R/3. Руководство для менеджеров и инженеров».

АРХИТЕКТУРА ВЫСОКОПРОИЗВОДИТЕЛЬНОЙ СИСТЕМЫ МНОГОАГЕНТНОГО МОДЕЛИРОВАНИЯ

О.Р. Набиуллин,
аспирант Нижегородского филиала Государственного университета—Высшей школы экономики,
nabiulin@tecomgroup.ru
В.М. Норкин,
ассистент Нижегородского Государственного технического университета ,
mnmvik@mail.ru

В первом разделе данной работы описывается область применения, современные тенденции и проблемы в области многоагентного моделирования. Акцент делается на производительность и распределение вычислительной нагрузки. Во втором разделе описывается подход к агентному моделированию и определяются базовые термины, которые будут использованы в дальнейшем. В третьем разделе предлагается архитектура фреймворка и среды для распределённого многоагентного моделирования. Прототип, созданный на основе предложенной архитектуры, рассматривается в четвёртом разделе. Анализ архитектуры и сравнение с альтернативными продуктами приводится в пятом разделе. В шестом разделе приведены основные выводы и обсуждение дальнейших перспектив развития предложенного прототипа.

1. Введение

Актуальность задачи

Существует множество областей профессиональной деятельности, изученность которых не позволяет использовать строгие математические модели. В качестве примера таких областей можно назвать некоторые виды экономических взаимоотношений, социальные процессы или задачи с экспоненциальной сложностью. Невозможность использования строгих моделей не означает их отсутствие. Препятствием может стать вычислительная сложность модели, отсутствие модели как таковой, или сложность подбора параметров. Для некоторого класса таких областей деятельности в качестве альтернативы выступает имитационное моделирование [1], позволяющее использовать синергетический эффект взаимодействия большого количества относительно простых систем для изучения систем более высоких порядков. Использование фреймворков для агентного моделирования позволяет специалисту сосредоточиться на предмете исследования, не вдаваясь в подробности реализации среды для проведения экспериментов. В идеале среда должна предлагать механизмы для построения и запуска модели и механизмы для контроля и анализа результатов эксперимента.

Проблемы производительности фреймворков для агентного моделирования

Многие фреймворки для агентного моделирования используют языки, близкие к скриптовым: Java (Swarm, RePast J, Quicksilver, VSEdit, MASON, JADE), Python (RePast Py), Lisp и его варианты (HLSIM, SimAgent) [2]. Скриптовые языки обеспечивают бóльшую абстрагированность и простоту, за счёт производительности. Разработчики таких фреймворков пытаются максимально снизить «планку вхождения» и позиционируют свои продукты как инструмент для непрофессиональных программистов. При этом проблемы производительности рассматриваются как второстепенные [3].

Тем не менее, рассмотрим возможность увеличения производительности путём распределения вычислительной нагрузки.

При построении распределённой агентной системы встаёт несколько проблем. Для примера рассмотрим стандартную модель «heat-bugs» (описание модели приведено в четвёртом разделе). Требуется, чтобы все агенты-жуки обрабатывались последовательно. Этот пример – частный случай более общей проблемы: наличие во многих агентных моделях глобальных разделяемых ресурсов (поле), доступ к которым должен быть синхронизированным.

Исследование [4] показало: визуализация результатов моделирования в системах Swarm и RePast замедляет работу в несколько раз, поскольку работа по просчёту (рендерингу) графики осуществляется на каждом шаге моделирования. Решение этой проблемы – разделение моделирования и представления, например, переход от прямой реализации к реализации основанной на событиях [5].

Предположим, что влияние разделяемых ресурсов минимально (т.е. к ним обращаются редко). В этом случае ограничивающим фактором является факт, что агенты, составляющие систему, должны обмениваться сообщениями. Стоимость такого взаимодействия сильно варьируется в зависимости от того, находятся ли агенты физически на одной машине или на разных. Временная стоимость посылки сообщения через Ethernet на 3–4 порядка выше, чем стоимость посылки локального сообщения.

Многие стандартные модели используют концепцию дискретного времени. Это означает, что эволюция системы осуществляется итерациями (шагами). Если мы попытаемся равномерно распределить агентов по некоторому количеству компьютеров, это приведёт к тому, что более быстрые машины будут простаивать.

Вышеперечисленные причины приводят к следующим выводам:

- ✧ если задаться целью максимизировать производительность, то это нужно делать средствами модели;
- ✧ использование C++ в качестве языка описания модели позволит увеличить производительность системы, не теряя в гибкости;
- ✧ распределение агентной системы имеет смысл если выполнены следующие условия:
  - ✦ отдельный «шаг» работы системы достаточно ресурсоёмок с вычислительной точки зрения;
  - ✦ объём передаваемых данных не велик;
  - ✦ модель предусматривает некоторое количество почти изолированных кластеров агентов (которые могли бы работать в рамках одной машины)

Распределённые вычисления

Распределение вычислительной нагрузки между несколькими копьютерами не обязательно должно быть «эквивалентным». Разделение может осуществляться по принципу: исходные данные – вычисления – представление результатов (визуализация).

Создание виртуальной распределённой среды взаимодействия

Для прозрачного взаимодействия агентов в рамках распределённой системы надо создать виртуальную распределённую среду взаимодействия. Аналогом такой среды в Java можно считать JMS (Java Messaging Service); в случае Windows-окружения похожую функциональность представляет MSMQ – Microsoft Message Queuing. В табл. 1 представлено сравнение основных характеристик этих подходов.

Таблица 1

Сравнение распределённых сред обмена сообщениями

Параметр	ASF (SOP)	JMS	MSMQ
Язык	C++	Java	Любой (CAPI, COM API)
Работа в гетерогенной среде	Да	Да	Да
Работа в offline	Нет	Нет	Да
Платформа	Любая	Любая	Windows
Лицензия	Свободная	Коммерческая (часть J2EE)	Коммерческая (компонент Windows)
Поддержка транзакционности	Нет	Есть	Есть
Защита от взлома	SSL	SSL	Encryption + Windows ACL
Акцент	Производительность	Распределённость и транзакционность	Гарантированная доставка

2. Подход

Реализация агентного подхода взята из SObjectizer [2], поэтому периодически в тексте будет фигурировать именно SObjectizer, а не ASF.

Агенты

В строгом смысле слова агенты в ASF (и в SObjectizer), не являются агентами, по определению из ИИ.

SObjectizer определяет агентно-ориентированную модель, в рамках которой осуществляется проектирование реализации конкретных задач. В рамках этой модели любое приложение рассматривается как совокупность именованных объектов – агентов. Каждый агент имеет заранее определённые состояния. Взаимодействие между агентами осуществляется посредством обмена сообщениями. Агент выбирает сообщения, которые он желает обрабатывать – подписывается на сообщения. Возникновение и последующая обработка сообщения называется событием. Сообщение может породить несколько событий. Они будут обрабатываться в соответствии с назначенными им приоритетами т.н. приоритетная диспетчеризация событий. Агенты, имеющие одинаковое множество состояний, событий и сообщений, образуют класс агентов.

Сообщения

Сообщение – единственный механизм взаимодействия между агентами в SObjectizer. Имя сообщения должно быть уникальным в рамках агента. На C++ сообщение должно быть представлено структурой или классом. Для отсылки сообщения SObjectizer предоставляет несколько функций send_msg. Сообщения могут выполнять роль сигналов (сообщение без данных, важен сам факт отсылки сообщения) и роль обычных сообщений (содержащих данные).

Для получения сообщения агент должен подписаться на него. Сообщения в SObjectizer идентифицируются по паре имён: имени агента, владеющего сообщением, и имени самого сообщения. Такой способ идентификации позволяет различать сообщения от агентов одного типа (владеющих одинаковыми наборами сообщений). Понятие «владение агента сообщением» возникает из-за необходимости как-то структурировать множество всех сообщений. Например, если в противопожарной системе есть множество однотипных датчиков, а каждый из них отсылает сообщение о своём состоянии, нужно как-то различать, к какому именно датчику относится конкретное сообщение. В SObjectizer эта проблема решена привязкой имён сообщений к именам агентов – если агент зарегистрирован в SObjectizer, есть и его сообщения. Как только агент дерегистрируется, вместе с ним исчезают и сообщения, которыми он владел.

Сообщения могут рассылаться адресно (т.е. с указанием имени конкретного агента-получателя), и широковещательно (без указания имени получателя). В случае широковещательной рассылки сообщение получают все подписавшиеся на него агенты. Сообщения могут доставляться получателям сразу, в момент отсылки сообщения (обычные сообщения), могут быть отложенными (т.е. их доставка будет производится после истечения некоторого тайм-аута) и/или периодическими (т.е. их доставка будет осуществляться в цикле с некоторой паузой между циклами).

События

Событие – это реакция агента на сообщение. Каждое событие должно иметь уникальное в рамках агента имя. На C++ событие реализуется нестатическим методом C++-класса агента.

Во время подписки на сообщение агент сообщает SObjectizer, что появление некоторого сообщения (далее – инцидент) приводит к генерации такого-то события. Событие может быть подписано на ноль или более сообщений. Если событие подписано на несколько сообщений, то событие генерируется при возникновении любого из инцидентов. Обработка события заключается в том, что SObjectizer автоматически вызывает метод-обработчик события при возникновении сообщения.

События имеют приоритет, устанавливаемый в момент подписки. Сообщение может быть инцидентом сразу нескольких событий одного агента, если эти события имеют разные приоритеты. В этом случае при возникновении сообщения обработчики событий запускаются SObjectizer в порядке убывания их приоритетов.

Пробинг (опрос состояния)

Встроенные средства SObjectizer позволяют для любого агента узнать его состояние. Список состояний, поддерживаемых агентом, декларируется на этапе создания. Для полноценного анализа информации о состоянии недостаточно. Предположим, что полное состояние агента описывается «древовидным» набором свойств. Тогда поддержка пробинга сводится к описанию в терминах сообщений и событий интерфейса, поддерживаемого всеми агенты, создаваемой системы. В качестве решения этой проблемы можно создать специальный тип агента discoverable_agent_t, от которого должны наследоваться все «пользовательские агенты». Наследование агентами свойств своих предков в SObjectizer похоже на наследование классов в C++, множественное наследование допустимо с ограничением: имена сообщений и состояний агентов-предков не должны пересекаться. Discoverable_agent_t должен определять

набор сообщений, позволяющий узнать у агента его внутреннюю структуру, полное состояние и статистические параметры.

Наблюдатели (аггрегаторы)

Вводится специальный тип агентов – аггрегаторы. Аггрегатор – это агент, находящийся в системе, но не оказывающий на неё активного влияния. Задача аггрегатора – аккумуляция статистической и аналитической информации в соответствии с определёнными правилами и предоставление их в виде сообщений. Например, в случае модели heat bugs в системе присутствует два вида аггрегаторов: unhappiness_aggregator, собирающий информацию о среднем уровне «несчастливости» агентов-жуков, и hb_gui_server – агент, генерирующий некоторое кодированное представление текущей ситуации на поле.

Асинхронность

Все сообщения, передаваемые агентами, асинхронные. Сколько потоков будут выполнять работу по обработке сообщений, зависит от диспетчера, который задаётся при старте SObjectizer. При использовании некоторых типов диспетчеров приложение может не заботиться о синхронизации доступа к разделяемым данным, т.к. самих разделяемых данных не будет. На текущий момент все штатные диспетчеры SObjectizer гарантируют, что несколько обработчиков событий одного агента не будут запущены в разных контекстах. Это означает, что у агента нет проблемы доступа к своим данным из разных потоков. Поэтому и нет необходимости синхронизировать доступ к ним.

Кодогенерация

Среда для агентного моделирования должна предоставлять возможность декларативного (графического) описания аггрегаторов и введения этих новых сущностей в текущий эксперимент.

Проверка гипотез

Одним из ключевых элементов отличающих разрабатываемую среду от аналогов является интерактивная проверка гипотез. Физически проверка осуществляется построением новых аггрегаторов и включением их в текущий эксперимент. Среда должна предоставлять удобные средства контроля и мониторинга аггрегаторов.

Сценарий работы исследователя со средой ASF следующий. Запускается эксперимент, в процессе взаимодействия состояние среды меняется, и аггрегаторы выдают некоторую информацию о протекающем процессе. В случает модели heat bugs одним из видов аггрегатора становится агент, вычисляющий среднюю «несчастливость» жуков на каждом шаге работы системы. Этот агент посылает свои сообщения, отображаемые средой на компьютере пользователя. В общем случае исследователь контролирует эксперимент, проходящий на другом компьютере или наборе компьютеров. Начиная с определённого этапа, у него возникает гипотеза: средняя «несчастливость» жуков не поднимется выше 0.016 но и не опустится ниже 0.015. Исследователь описывает на специальном DSL (Domain Specific Language) агента, проверяющего эту гипотезу, с двумя сообщениями: msg_avg_ok, которое должно генерироваться каждые сто шагов, если условие гипотезы не было нарушено и msg_avg_fail (val), которое должно генерироваться, если одна из границ оказалась неверной. После этого по описаню агента на DSL генерируется C++ описание агента (динамическая библиотека, содержащая сервис, который при запуске зарегистрирует агента в системе и создаст один экземпляр). Созданная библиотека распространяется средствами ASF по всем хостам, участвующим в эксперименте, и запускается. Предположим, в некоторый момент времени пришло сообщение msg_avg_fail (0.0161). Исследователь делает вывод, что гипотеза не подтверждается, и уточняет параметры. Теперь генерируется агент, проверяющий рамки 0.002, 0.015. Предыдущий агент исключается из эксперимента, и его место занмает новый с уточнёнными параметрами. Графически этот процесс отображен на *рис. 1.*

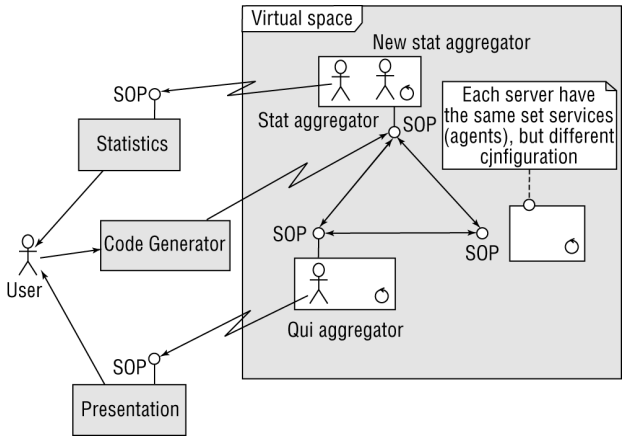


Рис. 1. Включение нового аггрегатора в эксперимент

Агентно-ориентированный подход – скорее дополнение к объектно-ориентированному подходу, нежели его замена. Фактически агентно-ориентированный подход является дополнением ООП в области динамического взаимодействия объектов, т.е.

в объектной модели разрабатываемой системы выделяются самостоятельные объекты, взаимодействующие друг с другом. Эти объекты объявляются агентами. Далее проектирование осуществляется практически так же, как и в ООП (описываются состояния, сообщения и события агентов), только с учётом того, что каждая из используемых сущностей (состояние, сообщение, событие) будет конкретным, заранее известным способом перенесена из проекта в программу.

3. Архитектура

После анализа работ [3, 7–10] выбрана архитектура, представляющая некоторое сужение SOA (Service Oriented Architecture).

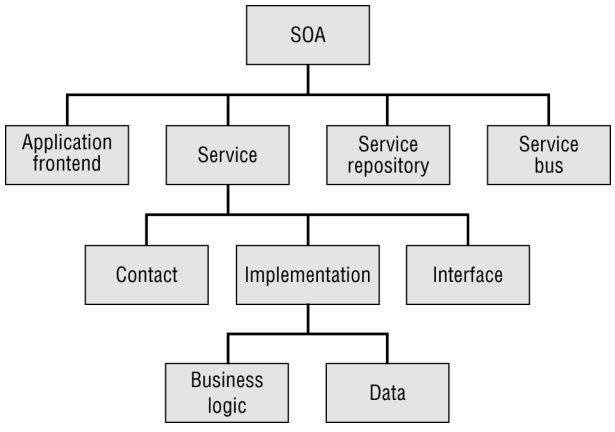


Рис. 2. Архитектура SOA

Один из ключевых аспектов проектируемой системы – его распределённость. Выбрана многозвенная (клиент-сервер) архитектура с одним важным дополнением: клиентская часть должна минимально отличаться от серверной. Этого удалось добиться в полной мере, выполняя различные задачи. Клиент и сервер содержат полный набор логики, что делает возможным смену ролей в runtime. При проектировании сформулированы следующие предпосылки:

- ✧ единая архитектурная концепция для разных приложений;
- ✧ модульность и расширяемость;
- ✧ быстродействие, близкое к Real-Time;
- ✧ принципиальная возможность перехода на другие программные платформы;
- ✧ самодиагностика;
- ✧ устойчивость к внутренним и внешним сбоям;
- ✧ защита информации и алгоритмов от несанкционированного доступа;
- ✧ конфигурирование «на лету».

Используемые библиотеки, технологии.

Standard C++

В качестве основного языка разработки выбран C++ как язык, обеспечивающий высокий уровень абстрагированности и позволяющий писать нативный эффективный код. [8, 11–14]. Обеспечение кросплатфоменности достигается использованием стандартного подмножества языка C++, а также инкапсуляцией платформенно-зависимых примитивов и сервисов. Большую часть инкапсуляции базовой функциональности операционной системы, такой как сокеты, потоки, динамические библиотеки, работа с переменными окружения, берут на себя библиотеки Boost и ACE.

Boost

Билиотека Boost – свободно распространяемый продукт с открытым исходным кодом (open-source) [15]. Исходники библиотеки и документация доступны по адресу www.boost.org [16]. Библиотека Boost – это собрание множества независимых библиотек, созданных независимыми разработчиками и тщательно проверенными на различных платформах. Можно считать, что Boost – это расширение стандартной библиотеки C++.

ACE

The ADAPTIVE Communication Environment (ACE) – одна из самых переносимых C++ библиотек, предназначенная для разработки сложных, многоплатформенных приложений, и широко используется во всем мире [17]. ACE представляет собой библиотеку, разделённую логически на несколько слоёв. На самом нижнем уровне находится кроссплатформенная реализация многих функций из стандартной библиотеки C и POSIX. Над ними – интерфейсные фасады, скрывающие детали реализации таких средств, как нити, процессы, примитивы синхронизации, различные механизмы межпроцессорного взаимодействия. Над ними каркасы.

Python

Python (па́йтон, пито́н) – мультипарадигменный язык программирования высокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных, такими как словари (хеш-таблицы), списки, кортежи. Большая часть реализаций, в том числе и референсная CPython, – интерпретаторы. Поддерживает классы, модули (они могут быть объединены в пакеты), обработку исключений, а также многопоточные вычисления. Питон обладает простым и выразительным синтаксисом.

Язык поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное и аспектно-ориентированное.

SObjectizer

SObjectizer – набор принципов, правил и ограничений на проектирование и реализацию программ в рамках агентного подхода, называемый агентной моделью; кросплатформенная OpenSource библиотека, http://sobjectizer.sourceforge.net, позволяющая реализовать приложение в терминах агентов, их состояний, сообщений и событий [18; 19].

4. Прототип

На основании проведённого исследования, а также изучения соответствующей аналитической литературы сформулированы следующие принципы проектирования:

- ✧ ядро – сервисы – Модель данных;
- ✧ настройка через внешние файлы;
- ✧ граф зависимостей между сервисами;
- ✧ несколько категорий сервисов;
- ✧ механизм Job Scheduler;
- ✧ использование интерфейсов и «перехватчиков» для вызова;
- ✧ шаблон Generic Attribute;
- ✧ единый сервис-контроллер для обработки запросов (конфигурируется через скрипт);
- ✧ транзакции;
- ✧ Service – Health – Monitoring в ядре;
- ✧ микрорестарт.

При создании прототипа выделено пять категорий сервисов:

- ✧ Boot-strap services – сервисы, в обязанность которых входит запуск и инициализация ядра;
- ✧ Kernel support services – сервисы ядра, сервисы реализующие основную часть функциональности framework (каркаса);
- ✧ Application support services – вспомогательные сервисы, аналог HAL – Hardware Abstraction Layer;
- ✧ Application services – сервисы специфичные для SensorMonitor;
- ✧ Service extensions – расширения сервера (plug-ins, script);

Состав каждой группы сервисов, для одного из проектов, созданных на ядре ASF, отражён на *рис. 3.*

Группы Application support services и Application services на *рис. 3* не содержат всех сервисов. В общем случае количество сервисов определяется приложением при запуске и в дальнейшем может изменяться.

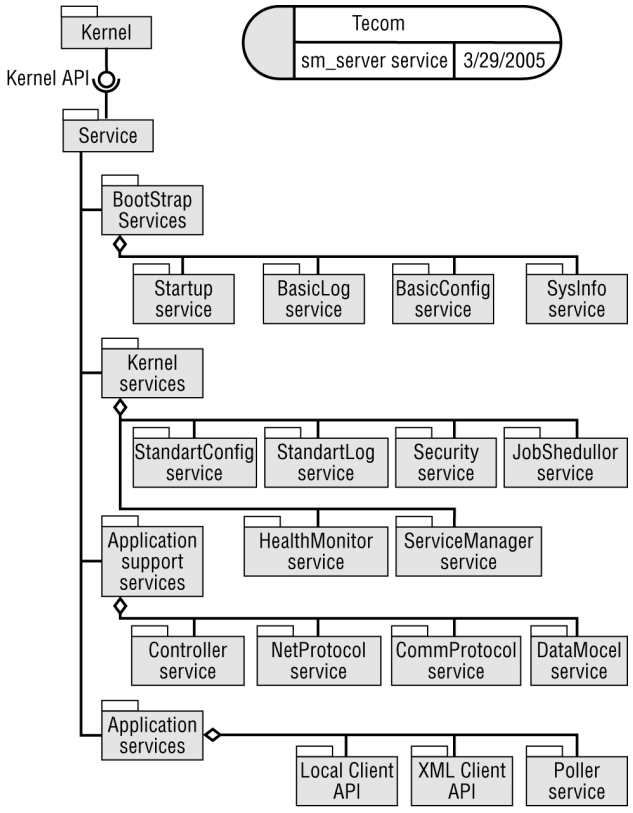


Рис. 3. Сервисы ASF

Ядро (Kernel) выступает в роли менеджера ресурсов (память, потоки, и т.д.) а также «черного ящика» для сервисов, через который они получают доступ к другим сервисам. Взаимодействие ядра с сервисами происходит асинхронно. Одна из ключевых особенностей – полная интроспекция: каждый сервис формально регистрирует свой внешний интерфейс в ядре. Таким образом, можно объединять сервисы, разработанные независимо друг от друга. В качестве средства оркестровки сервисов используется скрипт на языке Python.

Boot-strap services

Сервисы, входящие в группу Boot-strap отвечают за запуск ядра и базовую информацию о системе. Сюда входят:

- ✧ Startup service – сервис, которому передаётся управление при запуске сервера (экземпляр создается в функции main);
- ✧ BasicLog service – сервис, обеспечивающий журналирование (текстовый файл) на этапах запуска и остановки сервера;
- ✧ BasicConfig service – сервис, обеспечивающий доступ к конфигурационным параметрам.

- ✧ SysInfo service – сервис, через который ядро получает информацию о системе (количество доступной памяти, тип процессора и т.д.)

Сервисы этой группы тесно связаны с ядром, операционной системой и типом программы, играющей роль сервера в системе SensorMonitor (stand-alone, service, daemon), и любые изменения в их интерфейсах отражается на ядре.

Kernel services

Сервисы, входящие в группу Kernel реализуют основную функциональность framework (каркаса). Сюда относятся:

- ✧ StandartConfig service – сервис позволяющий другим частям системы получить доступ к своим конфигурационным параметрам (xml-файл);
- ✧ StandartLog service – сервис, обеспечивающий журналирование работы системы (текстовый файл, системные журналы, БД);
- ✧ Resource Manager – менеджер ресурсов обеспечивающий доступ к разделяемым ресурсам и их освобождение в случае сбоя сервиса, захватившего ресурс;
- ✧ JobScheduler – планировщик, обеспечивающий «одновременное» выполнение различных сервисов. Реализован как пул потоков (нитей);
- ✧ HealthMonitor – сервис, обеспечивающий мониторинг, завершение и перезапуск сервисов, в которых возникли ошибки (микрорестарт);
- ✧ ServiceManager – сервис, управляющий сервисами групп Application, и Application support, динамически подключаемыми на этапе загрузки.

Application support services

Сервисы, входящие в группу Application support, – прослойка между сервисами уровня приложения и аппратной платформой (операционной системой). Сюда относятся:

- ✧ Controller service – сервис, играющий роль драйвера Master-Controller;
- ✧ NetProtocol service – сервис, реализующий интерфейс к стандартныем сетевым протоколам;
- ✧ CommProtocol service – сервис, реализующий интерфейс в последовательному (RS-232) порту;
- ✧ DataModel service – сервис, представляющий собой модель предметной области.

Application services

Сервисы, входящие в группу Application services, реализуют фукциональность, специфическую для конкретного приложения. Сюда входят:

- ✧ SimulationManager – сервис обеспечивающий управление имитационным экспериментом;
- ✧ HeatBugsService – набор агентов для классической модели Heat-Bugs;
- ✧ HeatBugsGUI – визуализация модели HeatBugs (в т.ч. и распределенная);
- ✧ PythonScripting – управление приложением с помощью Python-скриптов.

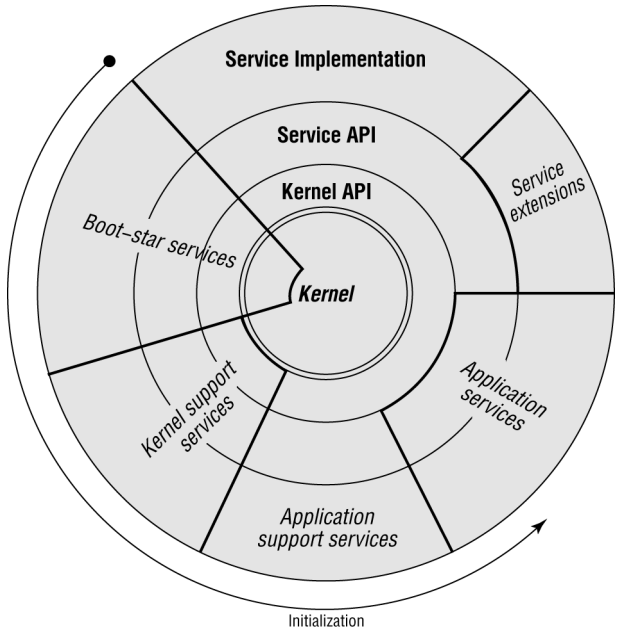


Рис. 4. Взаимодействие сервисов и ядра

SimulationManager

Сервис Simulation Manager, входящий в группу Application Services, обеспечивает поддержку проведения имитационных экспериментов. По сути этот сервис является интерфейсом между ядром ASF и SObjectizer. С точки зрения SObjectizer среда ASF представлена агентом system_agent, сообщения которого управляют ходом эксперимента. С другой стороны, для ASF эксперимент – это результат работы сервисов.

Важная особенность Simulation Manager – поддержка пробига. Класс discoverable_agent_t, базовый для всех агентов системы, реагирует на следующие сообщения:

- ✧ get_property_list_msg(string, string);
- ✧ get_property_msg(string, string);
- ✧ set_listener_msg(string, string);
- ✧ remove_listener_msg(string, string).

Первый аргумент каждого сообщения представляет собой «обратный адрес» отправителя. Строки аргументов, запросов и ответов содержат следующие специальные символы: «.,()\». Если строка

ответа должна содержать один из этих символов, они экранируются с помощью «\». Например: «ab.c» –> «ab\.c», «a\b» –> «a\\b». Все свойства строковые; бинарные данные передаются с помощью Base64-кодирования.

Сообщение get_property_list_msg(string, string) содержит два аргумента. Первый – «обратный адрес», второй – маска. Маска представляет собой строку вида «meters.temperature.m1» – спецификация полного имени свойства (ожидаемый ответ – пустая строка), или «meters.temperature.*» – спецификация частичного имени (ожидаемый ответ «meters.temperature.(m1, m2, m3)», если такие свойства существуют или «meters.temperature», если соответствующее свойство – лист дерева).

Сообщение get_property_msg(string, string) – запрос на получение значения свойства. Например, «meters.temperature.m1». Ожидаемый ответ в этом случае – «123». Или «meters.temperature.(m1, m2), meters.humidity.m1». Ожидаемый ответ – «(123,14),100».

Сообщения set_listener_msg(string, string) и remove_listener_msg(string, string) используются для подписки на изменение указанного в аргументе свойства агента. Структура сервиса Simulation Manager приведена на *рис. 5*.

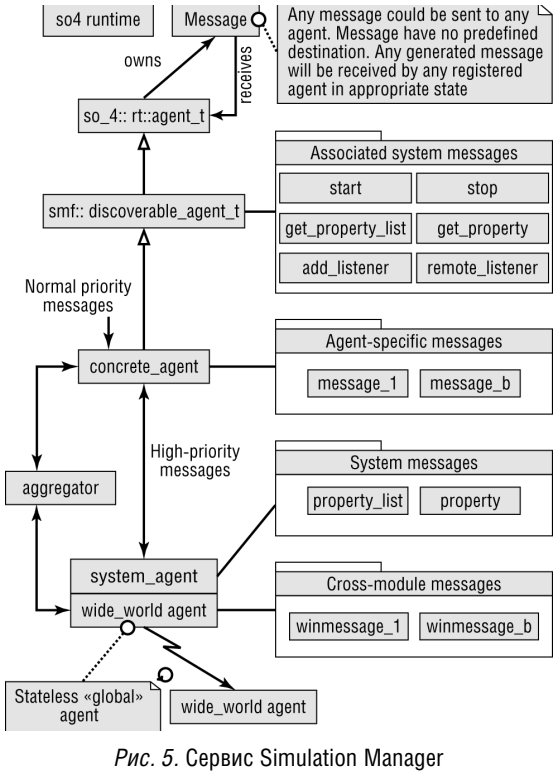


Рис. 5. Сервис Simulation Manager

С физической точки зрения все агенты представлены динамическими библиотеками, содержащими сервисы ASF. Будучи запущенным, такой сервис регистрирует в SObjectizer классы агентов. Создание экземпляров агентов осуществляется при вызове асинхронного метода сервиса. Такое двухфазное создание агентов призвано обеспечить гибкость при настройке системы и перезапуске эксперимента. Перезапуск происходит без перезагрузки (или даже остановки) сервисов. Соответствующая диаграмма развертывания приведена на *рис. 6*.

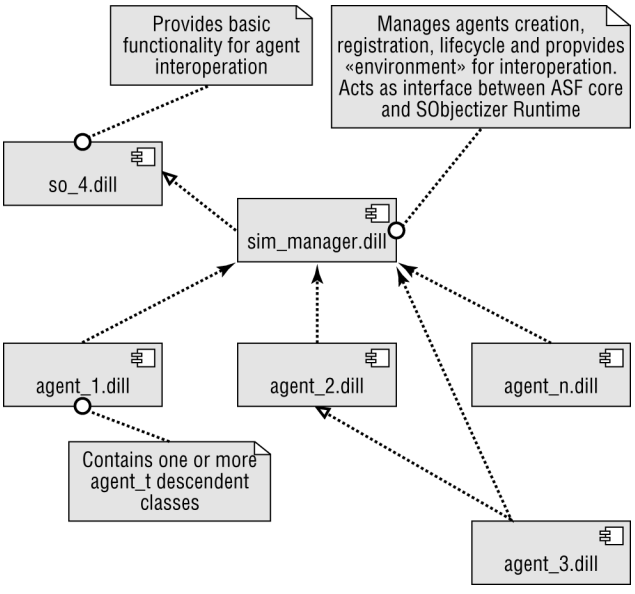


Рис. 6. Диаграмма развертывания для сервисов агентов

Принцип работы

Ключевой момент работы SObjectizer – диспетчеризация сообщений и событий. Механизм диспетчеризации события рассмотрен ниже.

Кто-то вызывает send_msg для отсылки сообщения. Функции send_msg указывается имя сообщения и, если необходимо, имя получателя. SObjectizer проверяет существование сообщения. Если такое сообщение действительно существует, создаёт экземпляр сообщения. В случае отложенного сообщения экземпляр сообщения отдаётся специальному объекту-таймеру для отсчёта тайм-аута. В противном случае начинается диспетчеризация сообщения. Для всех событий, подписанных на сообщение, формируется список заявок диспетчеру. Если сообщение отсылалось адресно, в список включаются только события агента-получателя. Заявки передаются диспетчеру для дальнейшей диспетчеризации. Диспетчер распределяет эти заявки по очередям заявок своих рабочих нитей. Например, диспетчер с одной рабочей нитью помещает все заявки в одну очередь. Диспетчер с активными объектами помещает заявки в очередь нити, обслуживающей конкретного агента.

Рабочая нить диспетчера извлекает из своей очереди очередную заявку. Она проверяет, может ли событие быть обработано в текущем состоянии агента.

Если может, то в контексте этой рабочей нити у объекта-агента вызывается метод-обработчик события. В противном случае событие игнорируется. После обработки очередной заявки сама заявка уничтожается. После уничтожения всех заявок, порождённых экземпляром сообщения, SObjectizer автоматически уничтожает экземпляр сообщения (если только сообщение не является периодическим).

Распределённое взаимодействие

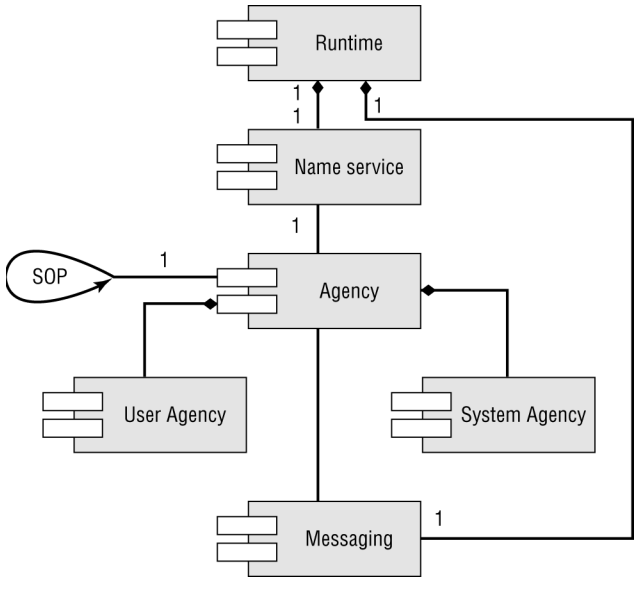


Рис. 7. Распределённое взаимодействие

В SObjectizer агенты взаимодействуют между собой только посредством асинхронных сообщений. Поэтому несложно обеспечить условия, при которых агенту-получателю неважно, откуда сообщение поступило. Необходимо только, чтобы сообщение содержало в себе копию всех необходимых данных. В этом случае сообщение может быть сериализовано, передано по какому-то IPC (Inter Process Communication) каналу в другой процесс, десериализовано и доставлено агенту-получателю. В SObjectizer для этого есть базовые средства.

В SObjectizer существует т.н. SOP (SObjectizer Protocol), определяющий, как будут сериализованы сообщения агентов. SOP позволяет передавать сообщения, которые содержат поля примитивных типов (char, int, short,...) или векторов этих типов, а также std::string и сложных объектов, сериализуемых с помощью ObjESSty (http://eao197.narod.ru/objessty). Для этого нужно всего лишь описать структуру сообщения для SObjectizer в виде набора макросов:

```
// Вот как эта структура описывает сообщение.
struct msg_alarm
{
// Имя датчика, инициировавшего данное сообщение.
std::string m_sensor_name;
// Текущие показания датчика.
float m_current_value;
// Предельное значение, после которого датчик объявляет тревогу.
float m_max;
};

// А вот так это сообщение должно быть описано для SObjectizer.
SOL4_MSG_START(msg_alarm, msg_alarm)
SOL4_MSG_FIELD(m_sensor_name)
SOL4_MSG_FIELD(m_current_value)
SOL4_MSG_FIELD(m_max)
SOL4_MSG_FINISH()
```

Такое описание позволяет SObjectizer на стороне отправителя взять значения полей m_sensor_name, m_current_value и m_max из объекта-сообщения и сериализовать их в коммуникационный канал. На принимающей стороне SObjectizer создаёт объект типа msg_alarm и восстанавливает значения его полей, затем доставляет созданный объект получателю.

SObjectizer вводит понятие коммуникационного канала (экземпляр конкретного IPC-соединения или сессии). Коммуникационные каналы обслуживают специальные транспортные агенты. Например, для организации взаимодействия двух процессов по TCP/IP в одном из них нужно объявить транспортного агента, который создаст серверный TCP/IP-сокет, а во втором процессе – транспортного агента, который создаст клиентский TCP/IP-сокет. Эти агенты будут устанавливать и обслуживать TCP/IP-соединения, объявляя каждую новую сессию отдельным коммуникационным каналом. Например, так выглядит создание транспортного агента для серверного сокета:

```
so_4::rt::comm::a_srv_channel_t a_channel(
«a_channel»,
so_4::socket::channels::create_server_channel(ip_address));
```

а вот так – создание транспортного агента для клиентского сокета:

```
so_4::sop::std_filter_t * filter = so_4::sop::create_std_filter();
filter->insert(a_common_t::agent_name());

so_4::rt::comm::a_cln_channel_t a_channel(
«a_channel»,
so_4::socket::channels::create_client_factory(ip_address),
filter,
// Вот этот объект будет указывать агенту повторять попытки
```

```
// установления соединения каждые 5 секунд, и инициировать
// повторное соединение после разрыва без задержек.
so_4::rt::comm::a_cln_channel_t::
create_def_disconnect_handler(5000, 0));
```

Над транспортными агентами в SObjectizer стоит еще один специальный агент-коммуникатор. Он отвечает за отслеживание всех доступных коммуникационных каналов и проверку их жизнеспособности. Одна из задач коммуникатора – пингование каналов, если в них нет активности, и принудительное закрытие канала в случае отсутствия ответов на пинги. Вторая задача коммуникатора – отслеживание сообщений, которые должны быть переданы в коммуникационные каналы, и восстановление сообщений, полученных из коммуникационных каналов. Какие именно сообщения коммуникатор будет передавать в коммуникационный канал, а какие игнорировать? В SObjectizer существует понятие т.н. глобального агента. Это агент, владеющий только сообщениями, но не имеющий ни состояний, ни событий. Он даже регистрируется в SObjectizer особым образом. Зато его сообщения коммуникатор отслеживает и рассылает в доступные коммуникационные каналы. Чтобы два процесса могли взаимодействовать между собой посредством SOP, им необходимо всего лишь зарегистрировать у себя одного и того же глобального агента (отсюда и его название, он как бы существует без оглядки на границы процесса). После этого любое сообщение глобального агента, порождённое в одном процессе, будет автоматически транслироваться в другой процесс.

Тестовая модель HeatBugs

Для демонстрации и испытания возможностей прототипа реализована модель HeatBugs – одна из классических демонстрационных моделей в агентно-ориентированных вычислениях.

Модель HeatBugs представляет собой прямоугольное поле, населённое агентами-жуками. Одновременно в одной ячейке поля может находиться не более одного жука. Каждый жук излучает в пространство некоторое количество тепла, которое с течением времени рассеивается в пространстве. Один жук не может обеспечить комфортной температуры для себя, поэтому он начинает движение в том направлении, которое ему кажется наиболее предпочтительным. Выбор предпочтительного направления осуществляется как выбор ближайшей к жуку незанятой точки, где температура ближе к идеальной, чем в текущем местоположении. Если много жуков сбиваются в одну кучу, вместе они так нагревают пространство, температура перестаёт быть комфортной, и жуки начинают стремиться уйти от кластера. Для каждого жука на каждом шаге моделирования определяется его «несчастливость», как функция от модуля разницы между текущей температурой среды и идеальной температурой, к которой данный жук стремится.

На *рис. 8* отображены два состояния, в которых может находиться агент-жук:

- st_initial – в нём жук находится сразу после запуска приложения и при перезапуске модели. В этом состоянии жук пытается найти незанятое место на поле, в котором он начнёт свои передвижения;
- st_sim_run – это основное и единственное состояние жука в течение эксперимента. Находясь в этом состоянии, жук пытается найти оптимальное место. Стоит обратить внимание, что параметры среды жук узнаёт у специального агента – a_heat_space; всё взаимодействие со средой – запросы и перемещения реализованы как посылка/приём сообщений.

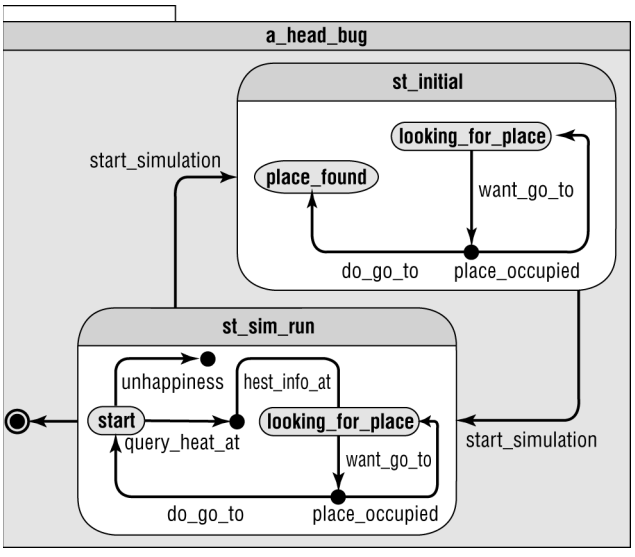


Рис. 8. Состояния агента a_heat_bug

Агент a_heat_space отображен на *рис. 9*. Агент может находиться в трёх базовых состояниях, привязанных к управлению экспериментом. В состоянии st_initial агент a_heat_space «размещает» агентов на поле. После успешного размещения всех агентов-жуков, модель переходит в состояние st_sim_run, где происходит основная часть взаимодействия в течение эксперимента. Состояние st_sim_paused введено, чтобы исследователь (или среда) мог в любой момент прервать эксперимент и просмотреть состояние отдельных агентов, участвующих в эксперименте.

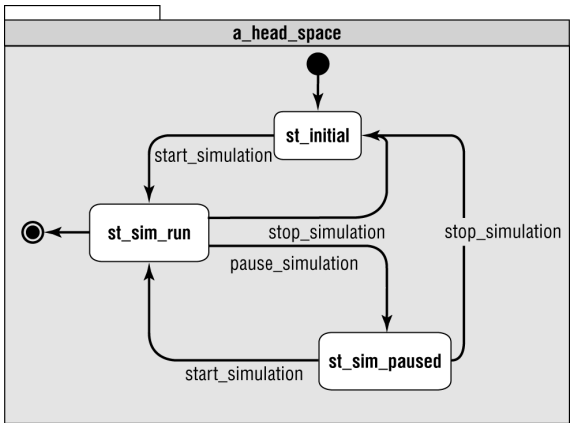


Рис.9. Состояния агента a_heat_space

Реализация графики

Одна из основных черт ASF – распределённость. В качестве иллюстрации этой концепции в прототип введён еще один сервис HeatBugsGui, обеспечивающий рендеринг графики. Этот сервис при запуске создаёт одного (или пару) из двух агентов: a_hb_gui_sender и a_hb_gui_receiver. Назначение этих агентов в сборе информации о текущем состоянии модели и отображении этой информации. Ни один из агентов не знает, запущен ли его компаньон локально, в одном адресном пространстве, или же работает на другой машине. Сообщения этих агентов помещаются на общую информационную шину и обрабатываются всеми, агентами, подписавшимися на эти сообщения. Если клиент, отображающий данные, запущен в другом процессе (и возможно на другой машине), работу по сериализации/десериализации сообщений этих агентов берёт на себя SObjectizer. Если модель запущена локально, никакой сериализации не происходит, и данные передаются практически прямым вызовом функции. Под локальностью понимается, что все агенты запущены в рамках одного процесса.

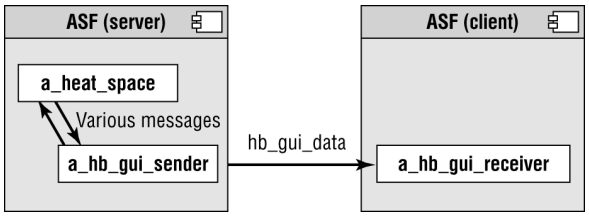


Рис. 10. Реализация удалённого GUI

5. Анализ архитектуры

Pro

Анализ работы прототипа показывает, что выбранная архитектура обеспечивает:

- ✧ **гибкость.** Приложение собирается как кубики из различных сервисов, причём переконфигурирование может происходить в runtime. Поддержка скриптинга позволяет держать систему «на кончиках пальцев», и формулировать запросы на языке высокого уровня (Python);
- ✧ **распределённость.** Создается виртуальная распределённая среда взаимодействия агентов. Важной отличительной особенностью является то, что агент, не знает и не должен знать, пришло сообщение к нему от агента находящегося с ним в одном адресном пространстве или с другой машины, возможно находящейся в тысяче километров от первой;
- ✧ **быстродействие.** Язык C++ позволяет воспользоваться преимуществами современных оптимизирующих компиляторов, генерирующий весьма эффективный код. Отсутствие сборщика мусора, характерного для управляемых (managed) языков, компенсируется использованием техник RAII, что позволяет осуществлять контроль не только за памятью, но и за другими ресурсами;
- ✧ **возможность «мягкой» интеграции с внешними системами.** При условии, что агенты должны взаимодействовать только сообщениями, мы обеспечиваем возможность мягкой интеграции с внешними системами. Например, на предприятии существует система поддержки принятия решений. Её часть – система имитационного моделирования, реализованная на ASF. С точки зрения ASF, интерфейсы к внешним системам – агенты, генерирующие свои сообщения в случае изменения внешних условий. Управляющие сигналы передаются во внешний мир через сообщения.

Contra

Было бы ошибкой утверждать, что выбранная архитектура лишена недостатков. Далее перечислены несколько проблем, к которым приводит следование архитектуре ASF.

Грамотный подход к выбору модели

Распределение вычислительных ресурсов может оказать эффект обратный, планируемому в случае моделей с большим уровнем связности. Если вычислительный эксперимент построен на десяти компьютерах объединённых сетью, и один шаг работы системы подразумевает получение реакции ото всех агентов, участвующих в эксперименте, причем каждый отдельных шаг вычислений достаточно прост, то накладные расходы по организации сетевого взаимодействия могут оказаться на несколько

порядков выше, чем сложность вычислений. Это приведёт к тому, что общее быстродействие снизится в соответствующее количество раз.

Внутренняя сложность (COM, Corba)

Внутренняя сложность фреймворка достаточно велика. Поддержка и развитее ASF требуют от программиста глубокого знания C++, а также специфики работы таких механизмов, как потоки, процессы, динамические библиотеки и сети. С точки зрения поддержки модульности и асинхронности, а также работы в распределенных средах, ASF частично реализует функциональность CORBA, обеспечивая динамическую переконфигурацию, и отказываясь от жесткой привязки к интерфейсам. Похожую функциональность в случае среды MS Windows обеспечивают дуальные COM-интерфейсы. Однако использование COM ограничило бы список доступных платформ для ASF. Использование C++ заставляет строго следить, чтобы настройки компилятора для среды и dll-расширений в точности совпадали, что делает возможным использование C++ интерфейсов через границу модуля [10][14].

Асинхронность усложняет реализацию

Для некоторых задач навязанная асинхронность может усложнить и замедлить реализацию. Однако реальный мир асинхронен. Если задача удобнее решается в синхронных терминах, имеет смысл рассмотреть альтернативы ASF.

Ненадежность соединения

Сетевые соединения ненадёжны. Поэтому прикладная логика модели должна содержать реакцию на выпадение одного или нескольких хостов из общей сети, что усложняет реализацию как сервера, так и модели.

Сравнение производительности

Чтобы получить численную характеристику выигрыша в производительности от использования ASF, проведено сравнение с распространёнными фреймворками для агентного моделирования Swarm и RePast (.NET).

Swarm

Тестовая модель HeatBugs с параметрами: количество жуков (агентов) 100, размер мира 100×100, количество шагов – 10000. ASF затратил на проход в среднем в 1,3 раза больше времени, чем Swarm. Это может показаться странным, но с другой стороны Swarm-модель использует прямые вызовы для определения температуры окружающего мира и выбора направления движения. Приблизительная оценка количества сообщений, обрабатываемых ASF и Swarm, показывает разницу на два порядка. Один шаг эмуляции в ASF-модели приводит к генерации и обработке примерно 500 сообщений, тогда как в Swarm – порядка 3-х. Стоит отметить, что модель HeatBugs – не самая удачная с точки зрения демонстрации преимуществ ASF, однако это классическая модель и авторы решили остановить свой выбор на ней.

RePast

RePast (.NET) на модели HeatBugs при параметрах, совпадающих с указанными в предыдущей главе, показал производительность в два раза меньшую, чем ASF. Отдельно стоит отметить, что модель RePast построена на прямых вызовах, что теоретически снижает накладные расходы на синхронизацию.

6. Выводы, планы

Созданная архитектура позволяет эффективно проводить имитационные эксперименты. Производительность системы намного выше, чем у аналогов, но выигрыш в производительности теряется в случае неудачно выбранной модели (в качестве мерила производительности используется количество сгенерированных и обработанных сообщений в секунду). Идеальный случай для ASF – модель, в которой небольшое количество агентов взаимодействуют асинхронно в силу природы модели, причём объём вычислений на каждом шаге эксперимента сравнительно высок, по сравнению с объёмом результатов вычислений.

Дальнейшее направление развития системы – создание DSL, позволяющего описывать агентов на высокоуровневом, графическом языке, и кодогенератора, для транслирования DSL-описания в C++ код. Помимо кодогенератора, в систему должен входить набор скриптов, содержащий настройки для компиляции расширений ASF, написаных вручную. ■

Литература

1. Gilbert, N. Terna P, How to Build and Use Agent-Base models in Social Science / N. Gilbert, P. Terna. // Mind & Society. 1999. № 1. P. [57–72].

2. Tobias, R. Evaluation of Free Java-libraries for Social-Scientific Agent Based Simulation / R. Tobias, C. Hofmann. // Journal of Artificial Societies and Social Simulation. 2004. № 7. P. [3–5].
3. Horling, B. The Soft Real-Time Agent Control Architecture / B. Horling, V. Lesser, V. Regis, T. Wagner. // Autonomous Agents and Multi-Agent Systems. 2006. № 12. P. [35–91].
4. Шутов, А. Создание распределенной системы экономического моделирования на основе RePast и Swarm / А. Шутов А, С. Капустин. 2006. 10 с.
5. Leow, R. Running C++ model under Swarm Environment / R. Leow, R. Standish. // 2005. 20 p.
6. SObjectizer. Online: http://sobjectizer.sourceforge.net/. 2008.
7. Fowler, M. Patterns of Enterprise Application Architecture / M. Fowler. Addison-Wesley Professional: 2002. 560 p.
8. Bulka, D., Mayhew D, Efficient C++ Performance Programming Techniques / D. Bulka, D. Mayhew. Addison-Wesley Professional: 1999. 336 p.
9. Bellas, F. An Agent-based Architecture for Building CORBA Distributed Systems. / F. Bellas, R. Juanes, N. Rodriguez, A. Viia. // Annual ACM Symposium on Principles of Distributed Computing. 1998.
10. Hofmeister, C. Applied Software Architecture / C. Hofmeister, R. Nord, D. Soni. Addison Wesley: 2000. 432 p.
11. Vandevoorde, D. C++ Templates: The Complete Guide / D. Vandevoorde, N Josuttis. Addison-Wesley Professional: 2002. 552 p.
12. Madina, D. A system for reflection in C++ / D. Madina, R. Standish // Proceedings AUUG 2001: Always on and everywhere. № 12. P. [207–207].
13. Jacobson, I. The Unified Software Development Process / I. Jacobson, G. Booch, D. Rumbaugh. Addison-Wesley Professional: 1999. 512 p.
14. Russian Software Developer Network. Online: http://www.rsdn.ru. 2008.
15. Karlsson, B. Beyond the C++ Standard Library: An Introduction to Boost / B. Karlsson. Addison Wesley Professional: 2005. 432 p.
16. Boost C++ libraries. Online: http://www.boost.org. 2008.
17. Huston, S. ACE Programmer's Guide, The: Practical Design Patterns for Network and Systems Programming / S. Huston, J. Johnson, U. Syyid. Addison Wesley: 2003. 544 p.
18. Охотников, Е. Соотношение SObjectizer и FIPA Abstract Architecture / Е. Охотников. 2006. 12 с.
19. Охотников, Е. SObjectizer: I Love This Game! / Е. Охотников. // RSDN Magazine. 2005. № 5. С. [19–21].

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ – ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

представляет свои периодические издания

ЭКОНОМИЧЕСКИЙ ЖУРНАЛ ВЫСШЕЙ ШКОЛЫ ЭКОНОМИКИ

ЕЖЕКВАРТАЛЬНЫЙ НАУЧНО-ИНФОРМАЦИОННЫЙ ЖУРНАЛ

Издается с 1997 г.

Главный редактор –
Евгений Евгеньевич Гавриленков

Журнал освещает теоретические и прикладные проблемы экономической науки. В каждом* номере – статьи ведущих российских экономистов. Рецензии, методологические и лекционные материалы. Эксклюзивные статьи зарубежных экономистов. Данные официальной статистики по широкому кругу вопросов.

Каталог Агентства «Роспечать» – индекс 79264
Объединенный каталог «Пресса России» – индекс 29233

Координаты редакции:
101990 Москва, ул. Мясницкая, 20, офис 235
e-mail: redact@hse.ru, тел./факс: (495) 628-0442

РОССИЙСКАЯ ЭКОНОМИКА: ПРОГНОЗЫ И ТЕНДЕНЦИИ

ЕЖЕМЕСЯЧНЫЙ СПРАВОЧНО-АНАЛИТИЧЕСКИЙ ЖУРНАЛ

Издается с 1993 г.

Главный редактор –
Елена Анатольевна Иванова

Журнал освещает состояние, динамика и дает прогноз основных социально-экономических индикаторов. В каждом номере – хроника событий экономики. Результаты конъюнктурных опросов предприятий. Самые свежие данные. Аналитический материал представлен с использованием таблиц, графиков и диаграмм.

Каталог Агентства «Роспечать» – индекс 79275
Объединенный каталог «Пресса России» – индекс 40548

Координаты редакции:
117312 Москва, ул. Вавилова, 7, офис 203
E-maii: id.hse@mail.ru Тел./факс: (495) 772-9571

РАСПРОСТРАНЯЮТСЯ ПО РОССИИ И ДРУГИМ СТРАНАМ СНГ. ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ – НА САЙТЕ: www.hse.ru

ИГРОВОЕ МОДЕЛИРОВАНИЕ КРИЗИСНЫХ ЯВЛЕНИЙ И СТРУКТУРНЫХ ПРЕОБРАЗОВАНИЙ В ЭКОНОМИКЕ

М.М. Крюков
к.э.н., старший преподаватель кафедры экономики природопользования экономического факультета МГУ имени М.В. Ломоносова
mikhailkryukov@yandex.ru

Рассматриваются методология и конкретные конструкции игровых имитаций социально-экономических процессов, в частности игра СТРАТЕГЕМА-1, которая с 1994 г. применяется на экономическом факультете МГУ.

Цель предлагаемой работы – показать, как в деловой игре отражаются процессы, порождающие нестабильность экономического развития, и пути её преодоления. Социально-культурный феномен игры тесно связан с критическими моментами развития, с точками, в которых нарушается пространственно-временная непрерывность событий. Игра проходит на стыке двух времён, знаменуя их смену, но также и преемственность. Сказанное относится и к деловым играм, точнее, тем из них, которые хранят память о своём игровом происхождении (в нормальном случае так и должно быть).

Такой игре присущ своеобразный драматизм, особое напряжение, вытекающее из внутреннего конфликта (коллизии), причём иногда даже не одного. По-настоящему удачной деловой игре не чужды категории завязки, развязки и кульминации, т.е. предельного обострения коллизий, приводящего к развязке. Отсюда следует, что кризисные явления в экономике и экологии могут рассматриваться как предмет первоочередного внимания для деловых игр.

Проведение деловой игры должно генерировать особый мир, живущий по собственным непростым закономерностям, подчинённый особой логике, не всегда заметной и понятной при взгляде извне. Если мы сумеем «прочитать» события, происходящие в этом игровом мире, они расскажут нам нечто важное и о действительности, лежащей за пределами игры. Чем сложнее и своеобразнее мир игры, тем более глубокие выводы можно из неё извлечь.

Требования к конструкции деловой игры кратко сформулированы ниже.

«1. В игре должно быть предусмотрено наличие нескольких ролей (или групп ролей) с различными, прежде всего экономическими интересами.

2. В игре должна быть выдержана пропорция между общением и процессами распределения ресурсов, т.е. она должна быть организована так, чтобы каждый контакт игроков осуществлялся по поводу движения ресурсов, а каждое изменение в материальной сфере, в свою очередь, требовало бы акта общения участников игры.

3. Деятельность участников игры должна оцениваться системой показателей, причём в их число следует включить обобщающие и агрегирующие характеристики работы каждого игрока, а также специфические игровые показатели, позволяющие оценивать его отношение к игре и отображаемой в ней действительности.

4. Игра должна отражать развитие системы во времени. К средствам достижения этого требования относится, в частности, организация отклика на принимаемые игроками решения»[1].

Создание новой имитационной игры – трудоёмкий процесс. Сначала следует поинтересоваться, нет ли уже готовой игры, отвечающей нашим запросам. Если мы и остановим своё внимание на каком-либо образце, он вряд ли удовлетворит нас полностью, поскольку создание игры было связано с конкретными обстоятельствами, не похожими на те, в которых работаем мы, и это накладывает на неё свой отпечаток. Неизбежна «пригонка» игры к нашим потребностям, – желательно, с минимумом вмешательства в её конструкцию. Следовательно, чтобы извлечь из игры возможности, существенные для нас, но первоначально скрытые даже для её авторов, в нашем распоряжении остаются методические средства.

[1]Крюков М.М., Крюкова Л.И. Принципы отражения экономической действительности в деловых играх. С. 44.

Для нас в течение длительного времени объектом интерпретации служила игра «STRATEGEM-1»[1], созданная под руководством Д. Медоуза (по-русски мы её называем СТРАТЕГЕМА–1). С 1994 г. она широко применяется в учебном процессе экономического факультета МГУ. Это не только дало богатый материал для анализа, но и позволило сильно продвинуть методику проведения игры. Речь идёт о выявлении, «прочтении» игровых событий и способах представления полученной при этом информации в ходе заключительного анализа (разбора) игры.

Данная работа – осмысление лишь очень небольшой части опыта, накопленного за годы использования этой игры.

Авторы игры СТРАТЕГЕМА–1 провозгласили своим рабочим принципом сочетание компьютерного моделирования с традиционными игровыми приёмами. К числу последних относится принятие решений в ходе непосредственного диалога игроков (а не в диалоге с компьютером). Поэтому в данной игре, как и в целой серии игр, разработанных под руководством или при участии Д. Медоуза, использована специальная игровая методика, объединяющая три элемента:

- ✧ относительно простую, но тщательно проработанную имитационную модель представленной в игре системы;
- ✧ портативный компьютер, способный работать от автономного источника питания, делающий «портативной» и саму игру;
- ✧ игровое поле с фишками, служащее вспомогательным средством учёта ресурсов, позволяющее визуализировать наиболее важные взаимосвязи в моделируемой системе и являющиеся центром общения участников игры.

Необходимости названных выше элементов – сама по себе существенный вклад в методику игрового имитационного моделирования. В особенности это касается функций игрового поля с фишками. Игру СТРАТЕГЕМА–1 характеризует продуманная ролевая структура, практически не позволяющая никому принять эффективное решение без обмена информацией с исполнителями других ролей. Не только центром такого общения, но и его средством, его языком становится игровое поле. Оно визуализирует, т.е. делает наглядной картину движения ресурсов. В результате получаем не просто баланс между общением и распределением ресурсов, но тесное переплетение, взаимопроникновение этих процессов. Всё это показывает, что рабочие принципы авторов данной игры и перечисленные выше наши принципы совпадают в существенных моментах.

Работа с игровым полем не позволяет игрокам ограничиваться одними интеллектуальными операциями, она постоянно выводит их деятельность во внешний, материальный план. Тем самым в процесс игры на равных правах включается двигательная активность. Следовательно, воздействию подвергаются все образующие элементы сознания.

Простота математической модели, заложенной в игру, не помешала отобразить в ней сложный комплексный характер взаимосвязей между экономической, социальной и экологической сферами жизни общества. Среди показателей, позволяющих оценить деятельность участников игры, присутствуют как специфически игровые, так и обобщающие, причём ни один из них не является решающим – только их совокупность даёт подлинную оценку. Правда, выбор обобщающих показателей оказался далеко не бесспорным. С его уточнением связано дальнейшее совершенствование методики игры.

Нами предусмотрен комплект материалов для её проведения, включающий игровое поле, набор фишек, компьютерную программу, ролевые инструкции для игроков, бланки игровых документов, описание игры для преподавателя, эскизы слайдов для вводного и заключительного занятий.

Нам довелось участвовать в первых экспериментах с игрой СТРАТЕГЕМА–1 в Советском Союзе, осуществлять научное редактирование одного из её изданий на русском языке. Полученный опыт принёс уверенность в целесообразности самого широкого применения её в курсе экономики природопользования на экономическом факультете МГУ. Одновременно стало ясно: многие положительные стороны игры пока только в потенциале. Чтобы их реализовать, нужно совершенствовать методику её проведения.

1. КРАТКОЕ ОПИСАНИЕ ДЕЛОВОЙ ИГРЫ «СТРАТЕГЕМА–1»

Участники игры разбиваются на команды. На каждую команду возлагаются функции «кабинета министров», управляющего развитием экономики условной страны. Предусмотрены пять «портфелей»: министров народонаселения, энергетики

[1]Д.Л. Медоуз, Ф. Тот. Стратиджем – 1 (Микрокомпьютерная обучающая управленческая игра о взаимодействиях между энергетикой и окружающей средой). Руководство по использованию. Riga: University of Latvia Ecological Center, 1993.

и энергосбережения, промышленности и социальной сферы, сельского хозяйства и охраны окружающей среды, внешней торговли и финансов. Одну роль могут исполнять один-два человека. В отдельном цикле игры моделируется пятилетний период развития экономики. Для проведения игры в полном объёме необходимо проиграть десять циклов.

Цель деятельности «правительства» – построить за отведённый срок высокоразвитое стабильное общество. Высокий уровень развития связан с достижением высокого уровня жизни населения.

Уровень жизни имеет в игре обобщённую числовую оценку, подсчитываемую по итогам очередного цикла. Она получается на основе показателей годового потребления продовольствия и промышленных товаров на душу населения, которые суммируются. Обеспеченность продовольствием берётся с весовым коэффициентом, равным четырём. Результат усредняется по всем сыгранным циклам. Данный индикатор рассматривается как количество «очков», набранное командой. Впрочем, следует понимать ограниченное значение итоговой балльной оценки и не слишком ориентировать участников на её максимизацию, ибо это может обеднить игру в смысловом отношении.

События в рамках одного цикла развёртываются следующим образом. Игроки сидят вокруг стола с игровым полем, на котором выложены фишки, изображающие доступные в этом цикле ресурсы. Они должны быть распределены по секторам поля, соответствующим направлениям использования ресурсов в различных секторах народного хозяйства. Хотя «министры» и объединены общей целью, каждый из них считает, что достижение этой цели зависит от подчинённого ему сектора экономики. Он старается получить как можно больше ресурсов, но они лимитированы. Команда находит пути разрешения возникающего конфликта, несмотря на всю его остроту. Это понятно: иначе она просто не сможет принять решение. В игре есть другие, более глубокие, более органические конфликты и противоречия, но они проявляются не сразу.

После того, как картина использования ресурсов окончательно сформирована на игровом поле, игроки фиксируют её в специальных листах решений и передают их преподавателю, ведущему игру, для обсчёта на компьютере. Математическая модель определяет последствия принятого решения: различные индикаторы и объёмы ресурсов, доступные в очередном цикле. Участники игры заново раскладывают фишки на игровом поле, отображая тем самым вновь полученные данные о положении своей экономики, и процедура повторяется.

Игровая экономика производит энергоресурсы, продовольствие, промышленные товары. Каковы направления их использования? Энергоресурсы можно экспортировать, расходовать на непроизводственное потребление населения, на производство промышленных товаров и продовольствия. Нераспределённый остаток энергоресурсов (другой продукции это не касается) хранится «до востребования». Продовольствие экспортируется и потребляется населением. То же относится и к промышленным товарам, но они ещё инвестируются в шесть видов капитала. Потребление промышленных товаров населением жёстко связано с потреблением энергии им же, так как на каждые пять единиц товаров необходимо дать одну единицу энергии.

В случае недостатка в ресурсах своего производства к услугам «кабинета» импорт. Источники его финансирования – экспорт и, при желании, кредиты иностранных банков, открывающие, в случае неосторожного пользования ими, перспективу долговой петли. Импортные цены на товары и продовольствие всегда выше экспортных цен (на энергоресурсы они равны).

Существенным является то обстоятельство, что ресурсы, произведённые внутри страны или закупленные за границей в текущем цикле, доступны для распределения только в следующем цикле.

Любой министр, ответственный за производственную деятельность, имеет в своём ведении два типа капитала. Для министра энергетики это капитал, занятый в производстве энергоресурсов, и капитал энергосбережения. Последний помогает снизить потребность в энергии на производственные нужды (выпуск продовольствия и промышленных товаров) до 30% от максимальной энергоёмкости. Темпы развития энергосбережения ограничены некоторой инерционностью: новые вложения в данную сферу не могут превышать суммы новых вложений в сельское хозяйство и промышленность. Если учесть, что исходная величина энергосберегающего капитала равна нулю и что в первых циклах инвестиционный потенциал невелик сравнительно с функционирующим капиталом энергоёмких отраслей, легко понять, что снизить потребность в энергии удаётся не так скоро, как хотелось бы.

Министру сельского хозяйства и охраны окружающей среды подчинён сельскохозяйственный капитал, определяющий производственные мощности по выпуску продовольствия, и капитал, вложенный в природоохранные мероприятия. Его функция – защита природы от вредных воздействий промышленности, энергетики и сельского хозяйства, но не восстановление уже причинённого

ущерба. Оно обеспечивается способностью среды к самовоспроизводству. Способность эта подчиняется закону, графически изображаемому колоколообразной кривой. По оси абсцисс откладывается показатель качества окружающей среды, меняющийся в пределах от нуля до единицы, а по оси ординат – улучшение этого показателя вследствие самовосстановления в процентах к текущему значению. Среда наиболее интенсивно воспроизводится в некотором среднем состоянии. При очень плохом и очень хорошем состоянии интенсивность регенерации среды падает.

Природоохранные мощности создают условия для наилучшего использования этой способности. Одна единица капитала, вложенного в них, нейтрализует последствия деятельности трёх единиц суммарного капитала в энергопроизводстве, промышленности и сельском хозяйстве. Недостаточная защищённость последней ведёт к ухудшению её качества, а это резко понижает продуктивность сельского хозяйства и повышает смертность населения.

Министр промышленности и социальной сферы контролирует вложения в мощности по выпуску промышленных товаров, в социальную сферу, которая в данной игре выполняет задачи здравоохранения и образования и весьма существенно влияет на качественные показатели роста и развития экономики.

Заложенная в компьютерную программу игры имитационная модель, основанная на методе системной динамики, включает сложные зависимости между распределением ресурсов и конечными результатами. Поэтому спектр возможных вариантов развития событий весьма богат. Соответственно ему может быть широка тематика, обсуждаемая в ходе собственно игровых занятий и во время послеигровой дискуссии. Она может охватывать, например, экологическое и экономическое значение энергосбережения и вообще ресурсосбережения, экономическую эффективность природоохранных мероприятий, проблемы энергетического кризиса и других кризисных явлений, связанных с переходом от индустриальной стадии развития общества к постиндустриальной.

2. НОВЫЕ ЭЛЕМЕНТЫ МЕТОДИКИ ПРОВЕДЕНИЯ ИГРЫ

Выводы из игры СТРАТЕГЕМА–1 тесно связаны с методом системной динамики, на котором основана математическая модель игры. Об этом позволяет судить как печатное руководство к игре, так и опыт участия в занятиях, проведённых Д. Медоузом во время его приездов в нашу страну. Метод системной динамики – инструмент моделирования и идеология, выйти за рамки которой автор игры полностью не смог или не захотел. Иногда создаётся впечатление, что авторская интерпретация игровых событий опирается не столько на анализ поведения системы в целом сколько на функционирование отдельных элементов модели. Много внимания уделяется экспоненциальному росту (в основном применительно к населению), с подробным разъяснением такого специфического параметра, как время удвоения. Акцент делается также на временные задержки в развитии некоторых процессов, на влияние срока службы отдельных капиталов и на предельную эффективность капиталовложений в те или иные виды капиталов, т.е. на дополнительную отдачу, приносимую каждой дополнительной единицей капиталовложений.

Картина поведения системы, как единого целого, не остаётся в стороне. Базой для демонстрации такой картины являются причинно-следственные диаграммы, показывающие взаимосвязи в главных подсистемах моделируемой системы. Подобные диаграммы – важный инструмент системной динамики, но они слишком сложны для восприятия в условиях ограниченности временем, отводимым на рассмотрение итогов игры, и для интерпретацию в свете этих итогов. Для обобщённой характеристики поведения системы более существенны наши выводы, касающиеся того, как в игре отражаются демографический переход, противоречия между долгосрочными и краткосрочными интересами, сравнительная эффективность факторов, прямо и опосредованно влияющих на один и тот же процесс.

Особое место в анализе итогов игры, по Д. Медоузу, занимает энергетический кризис, хотя в руководстве по игре сказано только о возможности его возникновения, видимо, для сохранения эффекта внезапности. Это действительно один из самых впечатляющих моментов игры, её кульминация. Его возникновение легко объясняется в терминах системной динамики как проявление нестабильности, свойственной системам с доминирующим контуром положительной обратной связи. Но ещё более важна естественность содержательной интерпретации данного события, позволяющей выявить его социально-экономический и экологический смысл.

Имитационная игра не должна быть средством пропаганды какой-либо теоретической концепции, в том числе и той, на которой основана модель отклика игры. Следует различать игру и процесс

исследования математической модели. Последняя важна в игре не сама по себе, а в качестве средства выражения. Вот почему в рассматриваемой нами игре недостаточно умения объяснять причины явлений на языке системной динамики. При анализе игровых событий надо уметь раскрывать их социально-экономический и экологический смысл. СТРАТЕГЕМА–1 даёт богатый материал для этого. Но при её разработке сделаны только первые шаги в нужном направлении.

Характеристика поведения социально-экономической системы в целом вряд ли будет полной без оценки эффективности её функционирования. Авторский замысел предусматривает упомянутую выше оценку предельной эффективности отдельных капиталовложений, а также оценку рентабельности отрасли по отношению ко вложенному в эту отрасль капиталу. Таким образом, речь не идёт о каких-либо интегральных показателях эффективности, как и о сравнении затрат и результатов, которые могут существенно расширить и углубить экономическую интерпретацию игры.

Любая деловая игра имеет свою терминологию, свой язык. Словарный запас рассматриваемой нами игры в её авторском варианте практически не включает такие экономические категории, как «национальное богатство», «валовой внутренний продукт», «национальный доход». Между тем макроэкономический характер игры делает применение этих понятий не просто естественным, но крайне желательным. Мы ввели в игру соответствующие показатели. Но этого недостаточно. Следует решить вопрос: как довести новую информацию до сознания участников игры, чтобы они не «захлебнулись» в её возросшем потоке?

При подготовке к разбору игры большой объём данных подлежит тщательному «просеиванию». Отбираются только главные факты, наиболее яркие и впечатляющие проявления закономерностей. Эту информацию необходимо сконцентрировать, чтобы изложить её в кратчайший срок (для игры СТРАТЕГЕМА–1 – одно занятие). Надо максимально использовать наглядность, что означает в данной ситуации обращение к графическому представлению материала. Опыт проведения СТРАТЕГЕМЫ–1 помог выработать оптимальный набор кривых, характеризующих динамику развития одной страны. Однако ограничиться простой их демонстрацией нельзя. Нужно уметь выделить на их основе наиболее существенные моменты, и обратить на них внимание участников игры. Все графики вместе дают известную гарантию, что эти моменты не будут упущены. Однако реальное обнаружение их через сопоставление динамики показателей одной или нескольких стран требует от преподавателя немалого искусства, отличного знания игры и затронутых в ней проблем экологии и экономики. Искусство разбора игры включает в себя умение проанализировать совокупность графиков данной страны (стран) в свете всей доступной информации по игре, привлечь наиболее интересные примеры из прошлого опыта и донести результаты этого анализа до игроков. Отсюда очевидна важность организации архива игры, накопления фонда информации по её проведениям. Графическое представление данных существенно упрощает поставленную задачу.

3. ЭНЕРГЕТИЧЕСКИЙ КРИЗИС

Довольно часто игра начинается при дефиците энергии и товаров для инвестиций. Консервируя очень низкий уровень потребления, играющим постепенно удаётся развить мощности по производству энергии и начать её экспорт. Полученная валюта тратится на капиталовложения в промышленность и энергетику. Игроки знают о важности энергосбережения, но не понимают в полной мере его значения. Первоочередной задачей на этом этапе они считают получение ресурсов, а не их эффективное использование. Любой избыток ресурсов снова пускается на производственные нужды. Постепенно отдача капиталовложений в энергетику падает. Тем более выгодно развитие энергосбережения в игре. Оно позволяет экономить ресурсы, повышает эффективность капиталовложений и уменьшает негативное влияние экономики на окружающую среду.

В начале игры «кабинет министров» вступает в управление страной с неразвитой экономикой, низким уровнем жизни населения, отсутствием социальной инфраструктуры и запущенной окружающей средой. По мере развития событий в игре происходит наращивание энергетических и производственных мощностей. К определённому моменту условная экономика приобретает типично индустриальную структуру, в которой добыча энергоресурсов имеет больший удельный вес, чем обрабатывающая промышленность. Перед игровым «кабинетом министров» стоит задача «перевернуть» структуру, т.е. перевести экономику своей страны в постиндустриальную стадию. Этого, конечно, нельзя сделать без перехода от экспорта энергии к её импорту, соответственно, от импорта готовой продукции – к экспорту. Но для этого необходимо увеличить долю потребления в национальном доходе, увеличить потребление энергии населением.

Переход от одной стадии развития к другой оказывается не гладким и не безболезненным. Здесь возможны нарушения стабильности, что выражается во внезапном энергетическом кризисе. Если его не преодолеть, налицо оказывается резкое падение производства и потребления, ухудшение состояния окружающей среды. Если выход из кризиса найден, условная экономика получает возможность осуществить переход к постиндустриальной стадии и устойчивому развитию.

Возникновение и развитие энергетического кризиса в игровой экономике иллюстрирует диаграмма «Эффективность роста энергетики» (*см. рис. 1, 2, 3*). Она включает следующие графики:

1. Капитал энергопроизводства, тыс. единиц.
2. Производство энергии, тыс. единиц.
3. Избыток (дефицит) энергии, тыс. единиц. Этот показатель рассчитывается как разность между наличием энергии к началу данного периода и её потреблением в этом периоде на производственные нужды и нужды населения.
4. Импорт энергии, тыс. ед.
5. Процент энергии, выделенной населению в данном периоде, к энергии, доступной к его началу.

При взгляде на эту диаграмму сразу бросается в глаза кривая избытка (недостатка) энергии. Вначале у всех стран лёгкий дефицит энергии. Затем, за редкими исключениями, он «рассасывается» в следующем же периоде благодаря вложениям в энергосбережение. Начинается постепенное нарастание избытка. Он становится резервом на будущее, но чаще – экспортируется и служит источником финансирования импорта других ресурсов, прежде всего промышленных товаров. Это нарастание длится несколько периодов.

Неожиданно, даже у самых благополучных стран, кривая избытка начинает «ползти» вниз, в большинстве случаев пересекая ось абсцисс и приобретая отрицательное значение. Возникает энергетический кризис.

Кризис может наступить в основном по двум причинам: недостаточное внимание к энергосбережению; отсутствие опережающего развития энергетики по сравнению с энергопотребляющими отраслями. Данная причина может вызвать кризис на ранней стадии игры. Когда перекосы, приведшие к нему, будут устранены, не исключено возникновение нового кризиса, уже по другой причине. Если кризис первого рода – следствие, в общем-то, просчётов, то кризис второго рода можно назвать структурным. Его избежать труднее, чем кризис первого рода. В его появлении его нет ничего удивительного и предвидеть его заранее вполне реально. Возможность кризиса второго рода заложена в структуру игровой экономики, согласно авторскому замыслу.

Внутренние энергетические ресурсы страны ограничены. Их может не хватить даже при полном

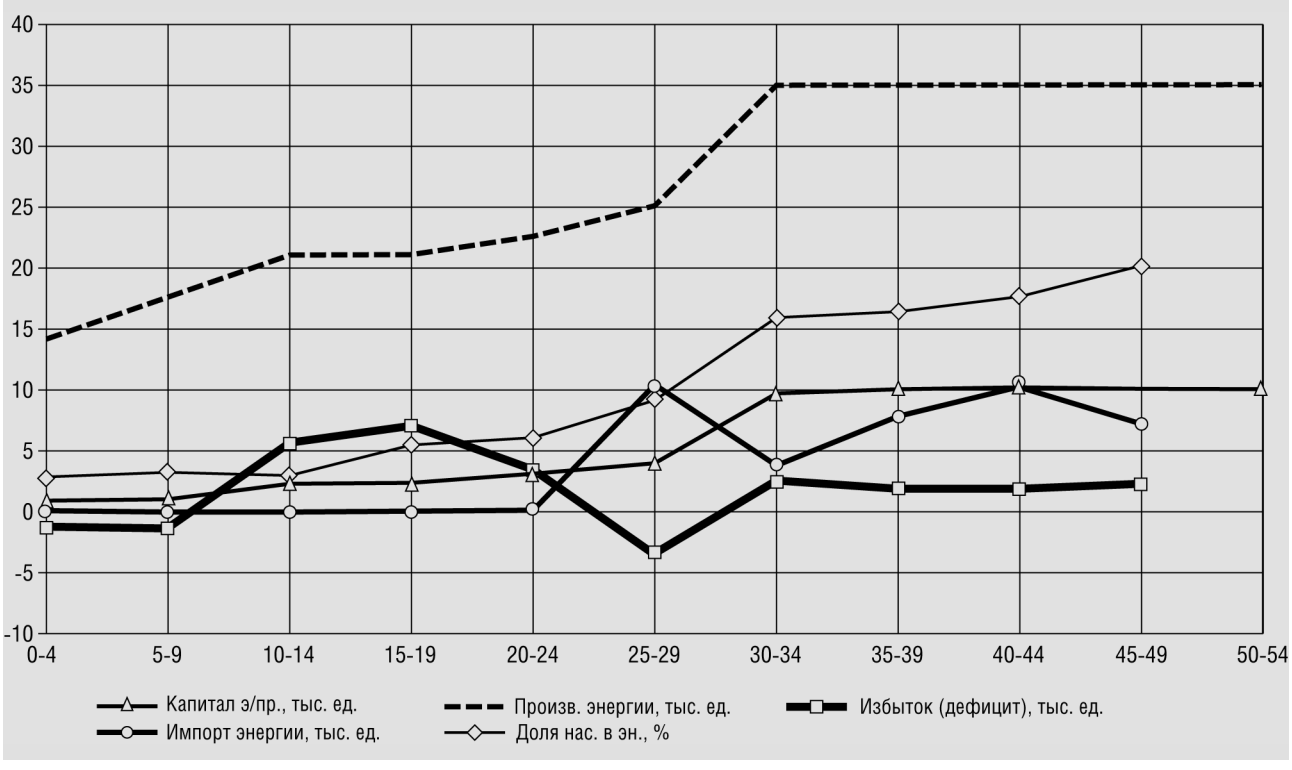


Рис. 1. Страна «Московия». Эффективность роста энергетики

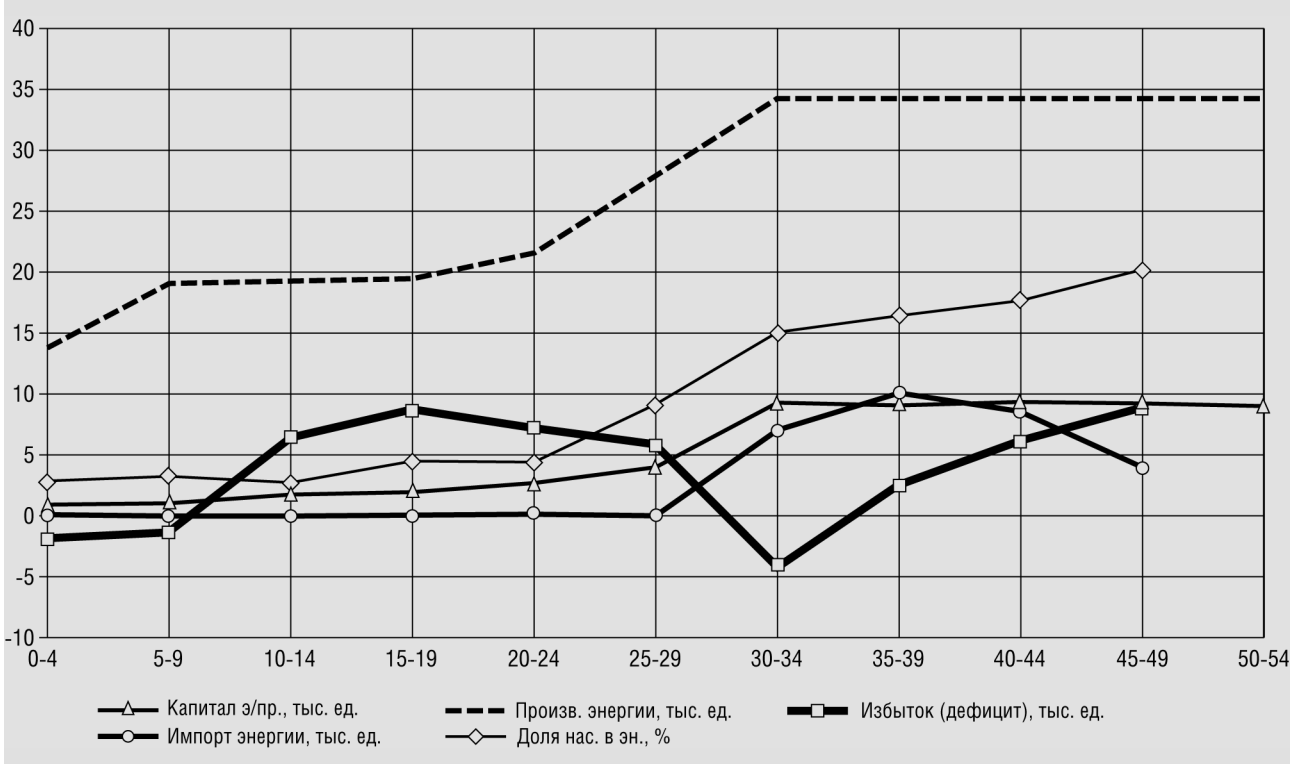


Рис. 2. Страна «Устиглин». Эффективность роста энергетики

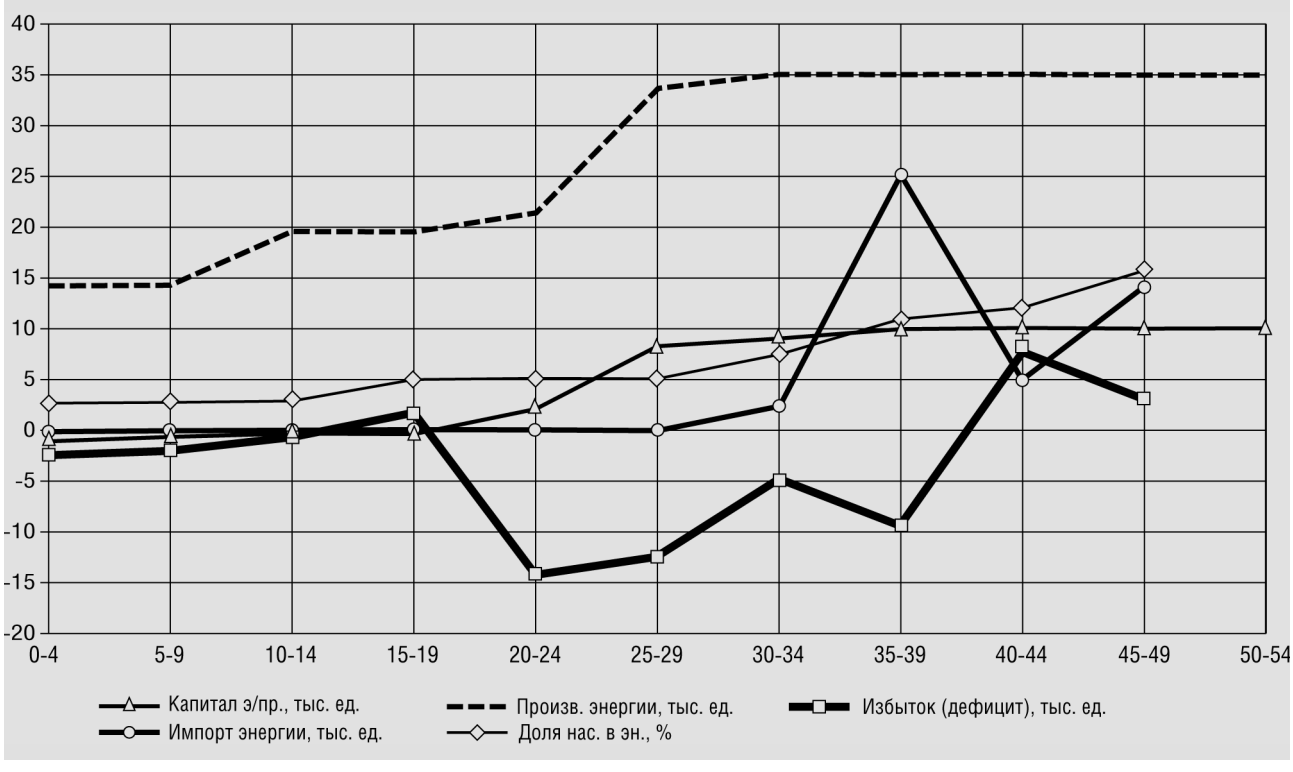


Рис. 3. Страна «Перцовка». Эффективность роста энергетики

развитии энергопроизводства и энергосбережения. Игроки же, уверенные, что они сделали всё для опережающего развития энергетики, теперь отдают всё внимание промышленности. По ликвидации энергетического дефицита недостаток ресурсов наиболее остро ощущается как дефицит капиталовложений, т.е. промышленных товаров. Промышленный капитал становится приоритетным направлением инвестиций (в какой-то момент это может привести к нарушению правила опережающего развития энергетики). Но если с энергетикой всё в порядке, если даже есть избыток энергии, то кажется, что есть и всё необходимое для развития промышленности. Отдача от инвестиций и от энергии, направляемых в промышленность, растёт. Каждый период в руках правительства игровой страны оказывается всё более значительный фонд промышленных товаров. Как его распределить? Прежде всего, нужно (с точки зрения игроков в данный момент) сделать новые вложения в промышленность. Тем самым налицо цикл обратной связи, причём положительной, что ведёт к нестабильности системы. Это выражается в том, что развивается такой промышленный потенциал, которому в один из будущих периодов не хватит энергии. Это отнюдь не уникальное, а весьма распространенное явление, потому что за ним стоят внутренние закономерности развития системы, а не особенности индивидуального стиля поведения участников игры, хотя и они оказывают своё влияние. Даже при максимальном энергосбережении потенциал собственного производства энергетических ресурсов, как правило, не может покрывать потребность промышленности и сельского хозяйства.

Момент доминирования положительной обратной связи и генерируемой ею катастрофы обычно обыгрывается во всех играх, основанных на системной динамике. Данная игра не исключение. С педагогической точки зрения она представляет собой один из наиболее удачных примеров использования этого момента как с точки зрения содержательной важности (как-никак, «энергетический кризис»!), так и в чисто психологическом аспекте (кризис действительно носит для большинства игроков, благодаря внезапности, катастрофический характер).

Все-таки, в отличие от других известных игровых системно-динамических моделей, здесь иногда успевает сработать управляющая отрицательная обратная связь. Это проявляется в том, что некоторые команды успевают заметить снижение избытка энергии и принять превентивные меры. В таком случае кризисные явления ограничатся падением избытка. Точнее сказать, вместо кризиса удастся обойтись предкризисными явлениями. Но что делать, если кризис уже налицо, или удалось обнаружить приближение предкризисных явлений?

Избыток энергии падает (или превращается в дефицит). Раньше этот избыток был источником экспорта. Значит, экспорт сырьевых (энергетических) ресурсов отныне прекращается или сокращается. Тенденция к падению избытка рано или поздно приведёт к дефициту. Проницательные игроки это понимают. Поэтому они думают уже не об экспорте энергии, а об её импорте. Откуда взять средства для импорта? Ответ один – перераспределить промышленную, отчасти и сельскохозяйственную продукцию.

Раньше промышленные товары нужны были преимущественно для инвестиций. Куда же направлять их теперь? В промышленность или сельское хозяйство? При дефиците энергии это напрасный перевод ресурсов, новое усугубление энергетического кризиса. Если мы не будем вкладывать средства в энергопотребляющие отрасли, то не сможем увеличивать и капитал энергосбережения. В принципе, настаёт время для восполнения некоторых пробелов. В частности, это самый подходящий момент для вложений в социальную сферу и в охрану природной среды. Это позволит сделать более эффективным использование промышленного и сельскохозяйственного капитала, следовательно, и дефицитной энергии, направляемой в эти отрасли.

Но если и после этого останутся нераспределённые промышленные товары, куда их девать? Конечно, их надо экспортировать, а на полученные средства закупать энергию, – или уже недостающую, или могущую стать дефицитом в одном из ближайших периодов. В последнем случае существенной оказывается возможность запасать энергию.

Но и импорт энергии часто не позволяет исчерпать наличные запасы товаров. Тогда остаётся один выход: увеличить (иногда весьма резко) потребление товаров населением. Правда, это однозначно требует выделения соответствующего количества энергии для населения. Подчеркнём: дефицитной энергии.

На диаграмме «Эффективность роста энергетики» рост кривой, отражающей процент энергии, выделяемой населению, в её общем наличии, как раз приходится на периоды, в которые ликвидируется или предотвращается энергетический кризис, – так же, как и рост импорта энергии.

Значительный рост доли энергии, потребляемой населением, в её суммарном потреблении страной наряду с переходом от экспорта энергетических и вообще сырьевых ресурсов к экспорту продукции

обрабатывающих отраслей (соответственно, к импорту сырья и энергоресурсов) – важнейшие признаки перехода к постиндустриальной стадии развития.

Таким образом, энергетический кризис «второго рода», т.е. структурный, а не связанный главным образом с просчётами в принятии решений, способствует переходу страны на постиндустриальную стадию развития.

Если правильно использовать энергетический кризис и руководствоваться в своих действиях интересами страны, можно решить сразу несколько важных социально-экономических задач, связанных с переходом страны на качественно иную стадию развития.

Возможны различные варианты действий в интересах страны. Можно, например, практически избежать энергетического кризиса, не развивая производственные мощности своей промышленности. Тогда не нужен импорт энергии и нет повода к сверхпотреблению. Другой вариант: промышленность развивается полностью, и её продукция идёт на экспорт. Энергия импортируется, а собственный энергетический потенциал используется не полностью. Тем самым как бы имитируется стратегия сохранения ресурсов для будущих поколений. Это прекрасный вариант устойчивого развития, отражаемый в игре.

В одном энергетическом кризисе могут сочетаться признаки кризиса первого и второго рода. Пример приведен на рис. 1. Для ликвидации энергетического кризиса, в котором оказалась страна «Московия» на 25–29 гг. своей деятельности, потребовалось не только двукратное расширение капитала энергетики (которая, следовательно, развивалась до этого несбалансированно с другими отраслями), но импорт энергии, и увеличение её потребления населением (так преодолевается кризис второго рода).

Пример одного только кризиса второго рода даёт страна «Устиглин» (рис. 2). Очевидно, кризис возник, когда капитал энергетики достиг своего максимума. Значит, причина кризиса не в недостаточном внимании «правительства» к развитию капитала энергетики, а в принципиальной ограниченности его величины. Пути преодоления кризиса были соответствующими.

Страна «Перцовка» (рис. 3) демонстрирует один затянувшийся кризис. Он явно распадается на две фазы. Первая из них (пик в 20–24 гг.) соответствует кризису первого рода, вторая (пик в 35–39 гг.) – кризису второго рода.

4. ЗАТРАТЫ НА ЕДИНИЦУ ПРОДУКЦИИ

Один из показателей эффективности функционирования игровой экономики (мы применили его и без него считал возможным обойтись автор игры) – затраты на единицу продукции. Нас интересуют преимущественно затраты на производство единицы промышленной и сельскохозяйственной продукции. Поэтому на соответствующей диаграмме приведены графики этих показателей, как и цена их импорта, практически всегда равная 1,1 (за исключением отдельных случаев). Это означает, что за единицу любого ресурса мы можем получить по импорту 0,9 единицы промышленного товара или продовольствия. Затраты на производство единицы этих ресурсов постепенно уменьшаются. Указанные затраты включают в себя расход энергии на нужды производства и амортизацию соответствующего капитала. Развивая энергосбережение, мы уменьшаем расход энергии на производство, тем самым сокращаем и затраты на единицу продукции. Это подтверждается тем, что на графике затраты падают почти параллельно с коэффициентом энергосбережения.

Что касается энергоресурсов, затраты на них складываются исключительно из амортизации. Если капиталоотдача в энергопроизводстве падает, то неудивительно, что затраты на единицу энергии растут, в отличие от других ресурсов. Но это не имеет большого значения. Начальная себестоимость энергоресурсов чуть больше копейки на рубль, в конце игры она доходит до 5–6 копеек. По сравнению с ценой импорта – это мизерные затраты. Самое главное – здесь мало что зависит от участников игры.

Гораздо интереснее проследить динамику затрат на производство в промышленности и сельском хозяйстве. Что бросается в глаза при рассмотрении этих показателей для большинства стран? Затраты на единицу продукции в первом периоде непомерно велики (в сельском хозяйстве почти три единицы, в промышленности – около двух).

Если себестоимость выше цены импорта, выгоднее импортировать ресурс. Источником финансирования импорта может быть экспорт энергоресурсов или кредиты. Если значение себестоимости заключено между ценой импорта и единицей, производство остаётся нерентабельным, но все-таки более выгодным, нежели импорт. Абсолютно выгодным собственное производство становится, когда затраты на единицу ресурса меньше единицы.

Лишь немногие из команд оказались способными быстро осознать эти закономерности. Ещё реже удавалось полностью реализовать вытекающие из

них преимущества. Когда в основу развития страны положены экспортно-импортные операции, трудно вовремя переключиться на собственное производство. Но самое сложное – правильно использовать временный выигрыш в ресурсах, полученный за счёт более выгодного импорта. Этот выигрыш представляет собой существенный источник инвестиционного потенциала страны. Но как следует его распределять?

5. СТРУКТУРА КАПИТАЛОВЛОЖЕНИЙ

В каждом периоде игры можно делать вложения в шесть капиталов:

1. Инвестиции в энергопроизводство.
2. Инвестиции в энергосбережение.
3. Инвестиции в промышленность.
4. Инвестиции в социальную сферу.
5. Инвестиции в сельское хозяйство.
6. Инвестиции в охрану природной среды.

Когда мы говорили о затратах на единицу продукции, фактически речь шла об эффективности использования энергоресурсов, поскольку они составляют большую часть затрат. Энергозатраты носят краткосрочный, разовый характер. А долгосрочная, стратегическая политика – это политика в сфере инвестиций. Структура инвестиций – важнейший управляющий рычаг в руках правительства игровой страны. От этой политики зависят и будущие затраты на единицу продукции.

Всякая инвестиционная стратегия, имеющая свое лицо, требует ресурсов. Если их нет, вряд ли можно сделать капиталовложения с существенными долгосрочными последствиями.

Анализ структуры капиталовложений для достаточного массива игровых стран позволяет сформулировать и проиллюстрировать ряд принципов капиталовложений.

1. Чем больше основная масса капиталовложений к концу игры, тем хуже. Это означает: развитие страны задержалось из-за недостатка источников капиталовложений или неэффективного использования ресурсов (на самом деле, две теснейшим образом связанные вещи), и основные капиталовложения делаются, когда они уже, в сущности, не нужны. Прекрасно, если капиталовложения «сдвинуты» к началу игры. Однако и здесь встаёт вопрос об источниках ресурсов. Если первоначальные источники нашлись, правильная политика капиталовложений в дальнейшем может стать источником ресурсов. И наоборот, неудачная политика лишит преимущества в ресурсах, которое удалось создать на старте.

Вообще же сдвиг капиталовложений к началу – почти идеальный и редко встречающийся случай. Нормально, если пик капиталовложений придётся почти на середину игры, но все-таки со сдвигом к началу на один-два периода. На протяжении нескольких последних периодов вложения в основные капиталы должны только восполнять их выбытие.

2. Инвестиции должны быть сбалансированы, т.е. не должно быть необъяснимого слишком очевидного отставания ни одной из шести позиций от других. Если в каких-то периодах это приходится делать, восполнять упущенное всё равно придётся. Тогда получится, что сначала приоритет отдаётся одной отрасли, затем другой, и т.д.

Подобное решение проблем развития страны по частям свидетельствует о недостатке ресурсов, а значит, о недостаточно эффективном их использовании. Впрочем, если уж нет другого выхода, лучше осознанное поэтапное решение проблем, чем метания от одной крайности к другой.

3. Вложения в энергопроизводство должны всегда опережать вложения в промышленность и в сельское хозяйство. Опережение здесь понимается в двояком смысле. Во-первых, принимается во внимание временной аспект. Добавочные энергоресурсы, добытые благодаря новым инвестициям в энергопроизводство, могут использоваться только на второй период после того, как капиталовложения сделаны. Во-вторых, новые вложения в промышленность и сельское хозяйство требуют и дополнительной энергии (а её наличие зависит от уже имеющегося капитала энергопроизводства и от коэффициента энергосбережения). Очень часто оказывается, что из-за отсутствия энергии вновь созданные капиталы в промышленности и сельском хозяйстве просто не могут работать.

4. Вложения в энергосбережение должны всегда стремиться к сумме вложений в промышленность и сельское хозяйство (больше этой суммы они не могут быть, а потому должны приближаться к максимуму). Если общая сумма капиталовложений незначительна, всегда лучше уменьшить другие вложения, и за их счёт увеличить вложения в энергосбережение.

5. В первые периоды максимум внимания должно уделяться вложениям в социальную сферу. Ей в это время отдаётся приоритет над энергопотребляющими отраслями. Вложения в энергопроизводство дают отдачу только на второй период, а реально воспользоваться ей можно только на третьем периоде. В первом периоде приходится обходиться 15 000 единиц энергии, во втором – 14 000 единиц. Чтобы заметно увеличить коэффициент энергосбережения, нужно сделать значительные вложения в промышленность

и сельское хозяйство. Хватит ли для них энергии даже при росте энергосбережения? Между тем вложения в социальную сферу увеличивают производительность труда в промышленности, следовательно, выпуск продукции без всякой дополнительной энергии и ухудшения качества окружающей среды. Итак, если есть потребность в работе собственной промышленности в первом – втором периоде, нужно обязательно в первом сделать вложения в социальную сферу. Она играет роль своего рода социальной энергетики.

6. ЭКОНОМИКА ИГРЫ «СТРАТЕГЕМА–1»

Детальный анализ изложенных выше аспектов игры позволяет прийти к определённым выводам относительно характера функционирования экономической системы, моделируемой в игре.

Цель достижения стабильно высокого уровня развития экономики и высокого уровня потребления ставится в условиях, весьма далёких от указанной цели. Причина этого не только и не столько в абсолютном дефиците ресурсов для выхода на высокие показатели, а в том, что имеющиеся ресурсы неэффективно расходуются, тогда как именно ограниченность ресурсов должна была бы «подтолкнуть» к более рациональному их использованию. Есть причины, которые не позволяют сделать это. Главная из них – заданная изначально структура игровой экономики. Преодолеть её, создать новую структуру, – задача, решение которой равнозначно достижению исходной цели. Но само такое преобразование – процесс, требующий средств, ресурсов. Если их источники не найдены, не продуманы заранее, структурная перестройка обратится в разрушение.

Характер распределения ресурсов в каждом цикле игры не оставляет сомнения в том, каковы могут быть упомянутые источники. Это: ограничение потребления населения на какой-то промежуток времени; прямое увеличение имеющихся ресурсов, преимущественно за счёт вложений в энергопроизводство и отчасти в промышленность; использование внутренних источников, что равнозначно тем же структурным преобразованиям. Совершенствование структуры экономики – это не только цель, но и источник ресурсов для достижения той же цели.

Но этот источник способен проявить себя только при условии, что есть какой-то стартовый минимум ресурсов – первоначальное накопление капитала. Откуда берётся этот минимум? Его источники не выходят за известные рамки. Это либо население, либо абсолютный прирост ресурсов, желательно за счёт внешних факторов. Таких факторов два: рост внешней задолженности и экспорт возможно большего объёма энергоресурсов. Последний в экстремальном случае может осуществляться за счёт полной остановки собственного производства – реализации некоторого подобия «шоковой терапии». Но подобные мероприятия имеют смысл и могут быть как-то оправданы только при условии, что они способствуют максимально быстрому преобразованию структуры экономики, переходу на внутренние источники развития. В противном случае дополнительные ресурсы, полученные из внешних источников, «ложатся» на старую структуру и бесследно исчезают в ней.

В конце игры в большинстве случаев так или иначе достигаются верхние (или разумные) пределы всех капиталов. Полученную структуру экономики можно считать рациональной. Но путь к ней лежит через неэффективную структуру, причём период её преобладания обычно слишком затягивается. Эффективная структура не успевает стать источником ресурсов. Значит, страна не сумела эффективно использовать свой главный ресурс – время, т.е. те самые 10 решений, отведённых на достижение цели игры.

Рассматриваемая нами игра предстаёт как целостный, сложный и неповторимый мир со своими внутренними закономерностями. В сложности игрового мира отражается реальный мир, реальная экономика с её закономерностями. ■

Литература

1. Крюков М.М., Крюкова Л.И. Принципы отражения экономической действительности в деловых играх. М.: Наука, 1988.
2. Д.Л. Медоуз, Ф. Тот. Стратиджем–1 (Микрокомпьютерная обучающая управленческая игра о взаимодействиях между энергетикой и окружающей средой). Руководство по использованию. Riga: University of Latvia Ecological Center, 1993.

Бизнес- информатика

Управление и бизнес в сфере информационных технологий!

– новое направление в России!

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Факультет бизнес-информатики
ВЫСШАЯ ШКОЛА БИЗНЕС – ИНФОРМАТИКИ

Программы бизнес-образования нового поколения:

ПРОГРАММА
«Мастер делового администрирования со специализацией в области бизнес – информатики» (MBA – IT)

Программа поддержана Ассоциацией Предприятий Компьютерных и Информационных технологий

ВТОРОЕ ВЫСШЕЕ
«Прикладная информатика в экономике»

ПРОГРАММА ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ
«Технологическое предпринимательство: управление инновациями в сфере высоких технологий»

ПРОГРАММА ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ
«Менеджер в сфере бизнес-информатики» (MBI)

Программы нового поколения, созданные ведущими специалистами в области бизнес-информатики, компаниями – лидерами в сфере ИТ, консалтинга и высоких технологий, качественно отличающиеся от имеющихся на рынке образовательных программ.

http://hsbi.ru, 772-95-61, 769-77-52