АРХИТЕКТУРА ФАБРИКИ ПРИЛОЖЕНИЙ

С.М. Авдошин,
к.т.н., профессор, заведующий кафедрой управления разработкой программного обеспечения, руководитель отделения программной инженерии факультета бизнес-информатики ГУ-ВШЭ
С.А. Белкин,
студент 2 курса магистерской программы «Управление разработкой программного обеспечения» факультета бизнес-информатики ГУ-ВШЭ

В работе предложен вариант построения архитектуры фабрики приложений, основанный на теории многоагентных систем.

Введение

Существует множество способов доступа и повторного использования внутренних и внешних знаний в компаниях. Как выбрать лучший из них? Существует ли он вообще? Как обеспечить правильный обмен и целостность знаний внутри компании, а также их тесную интеграцию? Как отслеживать эти процессы? Управление знаниями даёт ответ на эти вопросы. Многие исследовательские сообщества предлагают те или иные средства и решения, поддерживающие управление знаниями. Многие идут параллельным курсом и развивают отдельные направления в этой области. В будущем интеграция всего накопленного багажа знаний в области управления знаниями даст свои результаты.

С течением времени информационные технологии стали оказывать всё больше влиять на все области человеческой жизнедеятельности. Традиционные организации стали смотреть в сторону информационных технологий, наращивая объёмы данных, способы их хранения и обработки. Появились крупные *корпоративные информационные системы* (КИС), для работы с критически-важной информацией, и интегрирующие эту информацию в составе КИС. Конкурентная борьба между компаниями постепенно делает интеллектуальный капитал каждой из них стратегическим ресурсом, пользоваться которым большинство компаний не может, потому что не умеет.

Данная работа ориентирована на построение архитектуры *фабрики приложений* (ФП) активно взаимодействующей с базой знаний. Методология, реализуемая ФП, ориентирована на автоматизацию жизненного цикла производства *программного обеспечения* (ПО), посредством внедрения инновационных решений, основанных на концепции модельно-ориентированной [1] и компонентной разработки [2, 3].

К термину «фабрика приложений» не стоит относиться с большим скепсисом, заверяя, что превратив разработку в фабрику, мы лишим разработчиков желания мыслить, превратив их в роботов-рабов, рутинно-выполняющих комплекс поставленных архитектором задач. Всё с точностью наоборот. В современной производственной сфере ФП создает максимально-адаптированную среду, автоматизирующую рутинные задачи, и способствует фокусированию рабочих на решении креативных задач. ФП – это автоматизированная среда, ориентированная на максимальную автоматизацию производства продукта. Применительно к сфере информационных технологий концепция ФП ориентирована на максимум того, что можно получить при переходе к индустриально-ориентированному производству ПО.

Один из главных идеологов проекта по реализации концепции фабрик приложений – Джек Гринфилд определяет фабрики ПО следующим образом [4]:

«ФП – это согласованный набор процессов, средств и других ресурсов, используемых для ускорения всего цикла создания тех или иных программных компонентов, приложений или систем. Ускорение достигается за счет того, что специалисты получают руководства и рекомендации, в которых устанавливается последовательность действий, а именно, что нужно делать, когда, почему и как. Главенствующую роль играет формализация процесса: компоненты, которые могут быть быстро собраны и настроены, прототипы систем, которые можно быстро подготовить, а также специализированные средства, полностью или частично автоматизирующие механические рутинные задачи».

Мост между ФП и базой знаний – многоагентные технологии. С их помощью возможно добиться высокой степени интеллектуализации концепции ФП в отношении использования корпоративных баз знаний. Цель работы – построение на основе *многоагентной системы* (МАС) *системы управления корпоративной памятью* (СУКП), включающей базу знаний организации и интегрированной с развёрнутой в рамках организации (или одного из её отделов и департаментов) ФП.

1. Ключевой элемент организации – корпоративная память

Корпоративная база знаний или в общем случае *корпоративная память* (КП) – есть ничто иное, как целостное представление совокупности разнородных распределенных знаний организации, направленное на обеспечение эффективного использования сотрудниками при выполнении их собственных задач.

Крупные компании, имеющие за своими плечами не один десяток многомиллионных проектов, сталкиваются с неизбежной необходимостью использовать и учитывать опыт предыдущих лет. Знания, как и персонал, ресурсы и клиенты, также должны быть управляемыми и доступными в 99,9% случаев. Доступ должен быть эффективен, а результаты поиска – релевантны. Однако не стоит ограничиваться лишь средствами доступа к информации и эффективного извлечения данных из многочисленных КИС с их последующей структуризацией. Степень полезности знаний должна быть поднята на качественно новый уровень, т.е. знания должны использоваться программными экспертами, которые их оптимизируют и структурируют под конкретные нужды, будь то проектные, или общеорганизационные нужды. Это позволит достичь нового витка автоматизации деятельности организаций, в ходе которого, интеллектуальные интерфейсы, построенные с использованием знаний, смогут осуществлять большую часть рутинной деятельности, связанной с организацией рабочей среды, программных средств, процессов и прочих видов деятельности, всегда возникающих в каждом проекте. На наш взгляд, здесь много пересечений с теорией многоагентных систем.

Рассмотрим современную компанию, занимающуюся разработкой ПО. Среди документов такой компании неизбежно встретятся:

- ✧ инструкции, шаблоны, описания лучших практик;
- ✧ процессы;
- ✧ карточки проектов;
- ✧ учебные примеры;
- ✧ материалы по инженерным практикам (оценки и управления проектами, рисками, разработки и управления требованиями, и т.п.);
- ✧ материалы по технологическим экспертизам платформ (Java, .NET, Documentum);
- ✧ материалы семинаров;
- ✧ электронные библиотеки.

Это лишь базовый набор типов ресурсов КП, постоянно пополняющийся. Современные организации научились пользоваться и предоставлять доступ к такой информации.

Если мы рассмотрим компанию с различными департаментами, расположенными по всему миру, её база знаний может быть реализована в виде web-приложения, обеспечивающего всех сотрудников единой точкой доступа к распределенной внутри организации информации. Однако с чем вы сталкиваетесь, когда хотите найти описание какого-либо процесса, реализованного в организации? Что вы в таком случае делаете? Конечно, вы используете поисковые средства! Приложение отдаёт вам несколько сот совпадений с вашим запросом, а вы начинаете просматривать мегабайты документов. К ним относятся: текстовые документы, презентации, web-страницы, диаграммы, графики и прочие файлы, хранимые в КП. Следовательно:

- ✧ обмен информацией существует, однако ценная информация тонет в избыточности;
- ✧ поиск нелогичен или долог;
- ✧ интерфейс стандартен и не адаптивен.

Как же всё должно быть организовано? Ключевой фактор – адаптивность системы под нужды пользователя, обучение системы работе с конкретным лицом, анализ интересов. Допустим, вы участвуете в проекте, основанным на гибкой методологии разработки, и ваша команда разрабатывает крупную web-систему для корпоративного клиента. СУКП должна адаптировать среду взаимодействия проектных членов под конкретный проект. В расчёт должны приниматься опыт участников проектной команды, их предыдущие заслуги, опыт взаимодействия с самой СУКП. Система должна предоставлять каждому участнику рабочего процесса доступ к:

- ✧ спецификациям;
- ✧ планам итераций;
- ✧ практикам;
- ✧ регламентам;
- ✧ необходимым документам и книгам;
- ✧ схожим проектам;
- ✧ тренингам.

Интеллектуализация информации внутри СУКП позволит сотрудникам использовать хранимые знания интенсивнее и эффективнее. Естественным продолжением такой интеллектуализации станут интерактивные возможности. Пользователи смогут общаться с системой на новом уровне:

- ✧ сколько в компании сертифицированных специалистов по Mercury;
- ✧ сколько у компании было проектов на .Net Framework v 1.2;
- ✧ где взять кейсы по нагрузочному тестированию;
- ✧ какой у компании опыт тестирования методом белого ящика;
- ✧ где взять описание гибкого процесса разработки, принятого в компании;
- ✧ где взять пример документа спецификации программных требований для отправки заказчику;
- ✧ какие продукты и практики используются для улучшения качества кодирования;
- ✧ где взять описание процесса открытия нового выделенного центра;
- ✧ кто имеет опыт работы с Documentum;
- ✧ какие лицензии на продукты Rational есть у компании;
- ✧ какие тренинги по ProjectServer были разработаны сотрудниками компании.

Отметим свойства корпоративной памяти:

- ✧ КП – гетерогенная распределенная информационная среда. Равно как Web-среда, КП «испытывает» проблемы с точностью представления данных, т.к. по своей сути всякая информационная среда состоит из разнородных ресурсов. Можно полагать, что КП есть корпоративная семантическая паутина [5], в которой за описание корпоративных документов отвечают аннотации. Аннотации (как пример метаданных) играют ключевую роль в рассмотрении вопроса о распределённых знаниях;
- ✧ пользователи КП гетерогенны и распределены внутри организации. Задача СУКП – обеспечение пользователям интерфейса взаимодействия с корпоративной памятью, а также возможности участия в пополнении и расширении КП;
- ✧ задачи, выполняемые над КП, также гетерогенны и распределены. Пользователи и сама КП гетерогенны, поэтому применение МАС становится очевидным. МАС задают новый уровень абстракции в моделировании и разработке распределенных систем [6, 7]. С одной стороны, агенты способны адаптироваться к прикрепленным пользователям и ресурсам, с другой, совокупность кооперированных агентов может решать целый комплекс задач над КП.

В действительности, чтобы управлять такой структурой, как КП, стандартных средств недостаточно. Необходимо обеспечить систему механизмами адаптивности и реактивности, целеполагания, самоорганизации. На помощь приходят идеи многоагентных систем, состоящих из совокупности агентов, помещенных в некоторую среду и взаимодействующих между собой с целью достижения общих целей.

Рассмотрение многочисленных аспектов СУКП выходит за рамки данной статьи. Однако важно понимать, что в общей схеме взаимодействия между ФП и КП неизбежно возникает слой МАС, который входит в обе части, объединяя их воедино. Таким образом, достигается возможность использования данных не только самими пользователями, но и агентами, автоматизирующими процесс разработки программного обеспечения.

Суть построения информационной СУКП заключается в когерентной интеграции распределенных знаний корпорации, а также в пополнении знаний, обмене знаниями, их сбережении. КП по аналогии с человеческой памятью позволяет пользоваться предыдущим опытом и избегать повторения ошибок, КП фиксирует информацию из различных источников предприятия и делает эту информацию доступной специалистам для решения производственных задач. КП не позволяет исчезнуть знаниям выбывающих специалистов (уход на пенсию, увольнение и пр.).

2. Базовые концепции ФП

Разработка фабрики основана на четырех базовых понятиях: линейка программного обеспечения; ресурсы и активы; база знаний фабрики; модельно-ориентированная разработка.

С термином «линейка» мы сталкиваемся в повседневной жизни, начиная от множественных классов автомобилей и заканчивая линейками моделей телевизоров, мониторов, компьютеров, телефонов и т.п. Линейка ПО – это набор программных систем, объединенных схожим набором целей, функций, сегментацией на рынке, которые разрабатываются с помощью одинаковых активов. Линейка покрывает конкретизированную область, например, область разработки систем управления взаимодействия с клиентами или систем планирования ресурсов предприятия.

Ключевые атрибуты линейки.

- ***сфера использования*** – описывает то, какие продукты могут быть разработаны в рамках данной линейки;
- ***изменчивость*** – определяет некоторый ограниченный изменчивый набор функций продуктов;
- ***расширяемость*** – определяет «точки расширения», которые могут быть использованы для расширения функциональности продуктов, существующих в рамках линейки. Расширяемость включает функциональность, которая может быть использована, однако не подпадает в основную область применения.

Разработка линейки задает некий общий подход к анализу требований, дизайну и разработке гибкой архитектуры семейства продуктов. Владелец такой линейки вправе задать любой набор используемых активов для реализации определенной функциональности продукта.

Среди многократно используемых ресурсов и активов выделяют один наиболее важный – архитектурный каркас – отправную точку, или базовую площадку, относительно которой может быть разработан любой продукт в рамках линейки. Каркас покупается или разрабатывается самой организацией, разрабатывающей программный продукт в рамках фабрики. Разработка каркаса заключается в аккумулировании всех ресурсов организации: классов, компонентов, конфигураций, образцов и т.п. Объединение всех активов в рамках каркаса позволяет повысить уровень абстракции при разработке новых концепций и систем. Активы тщательно отбираются на стадии разработки фабрики на основе линейки ПО. Подробное описание линеек и подходов к их разработке можно найти в [8].

База знаний фабрики включает различные руководства, справочники, статьи, примеры программ и программы-образцы (паттерны) [9]. В общем случае рекомендация – это любая форма помощи для разработчика.

Модельно-ориентированная разработка, заключается в использовании моделей для создания элементов программного обеспечения [1]. В частном случае это генерация кода по абстрактной модели, оперирующей некоторым набором понятий, относящимся к выбранной предметной области, вокруг которой строится линейка продуктов. Здесь рассматриваются языки предметной области DSL [10], а также графические дизайнеры или конструкторы, цель которых – моделирование понятий в заданной предметной области. Цель данного моделирования не столько в отражении основных концепций, сколько в возможности генерации кода по заданным спроектированным моделям. Модели могут быть валидированы, с их помощью можно визуализировать различные уровни проектирования продукта, проводить конфигурирование и анализ. Модельно-ориентированная разработка позволяет:

- увеличить уровень абстрагирования, моделировать и управлять разработкой наиболее важных частей, скрывая детали реализации;
- связывать дизайн с кодом.

Таким образом, модели становятся наиболее важным артефактом в рамках разработки проекта.

3. Схема и шаблон – ключевые составляющие ФП

Процесс создания и реализации фабрики схож с процессами разработки обычных программных проектов, и является итерационным. Однако создание фабрики не может быть начато на пустом месте. Предполагается, что компания, в среде которой будет развертываться фабрика, уже обладает достаточными знаниями, а точнее активами. Интегрирование фабрики имеет смысл только в том случае, когда бизнес компании достаточно развит.

Опыт создания отдельных продуктов компании претерпевает ряд изменений, структурируется и приобретает требуемую форму линейки *(рис. 1)*. Затем весь багаж знаний преобразуется в некую глобальную модель или схему фабрики, охватывающую все присутствующие активы, точки зрения и связи между ними. В зависимости от требований к разработке того или иного продукта из схемы извлекается шаблон фабрики. Он уточняет набор требуемых ресурсов и автоматизирует работу *интегрированной среды разработки* (ИСР) ПО. С течением времени активы фабрики будут неизбежно пополняться, фабрика будет «разрастаться».

Как правило, фабрика *(рис. 2)* рассматривается:

- с позиции того, кто имеет прямое отношение к разработке самой фабрики, т.е. разработчика фабрики;
- с позиции разработчика решений на основе экземпляров фабрики, т.е. пользователя фабрики.

С обеих позиций фабрика – это ресурс. Только в случае разработчика фабрики, он представляет собой коллекцию избранных и тщательно подобранных активов и рекомендаций, позволяющих реализовывать полноценные приложения в рамках линейки. С другой стороны, фабрика интегрируется в среду разработки пользователей фабрики, позволяя

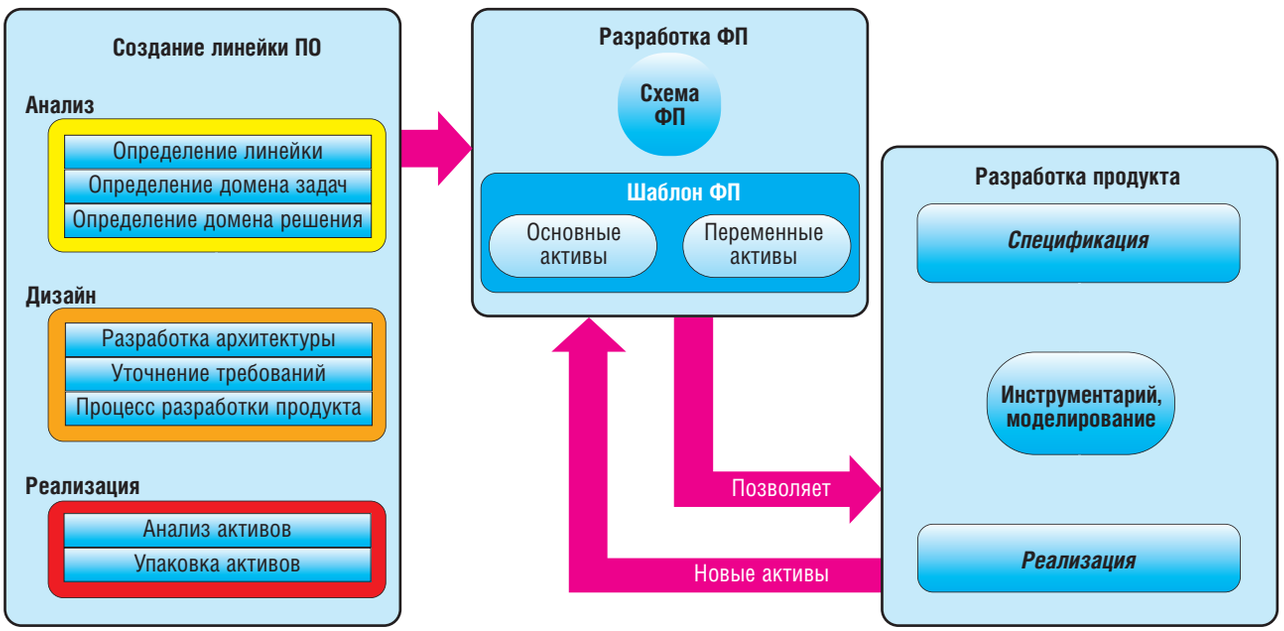

Рис. 1. Схема построения ФП

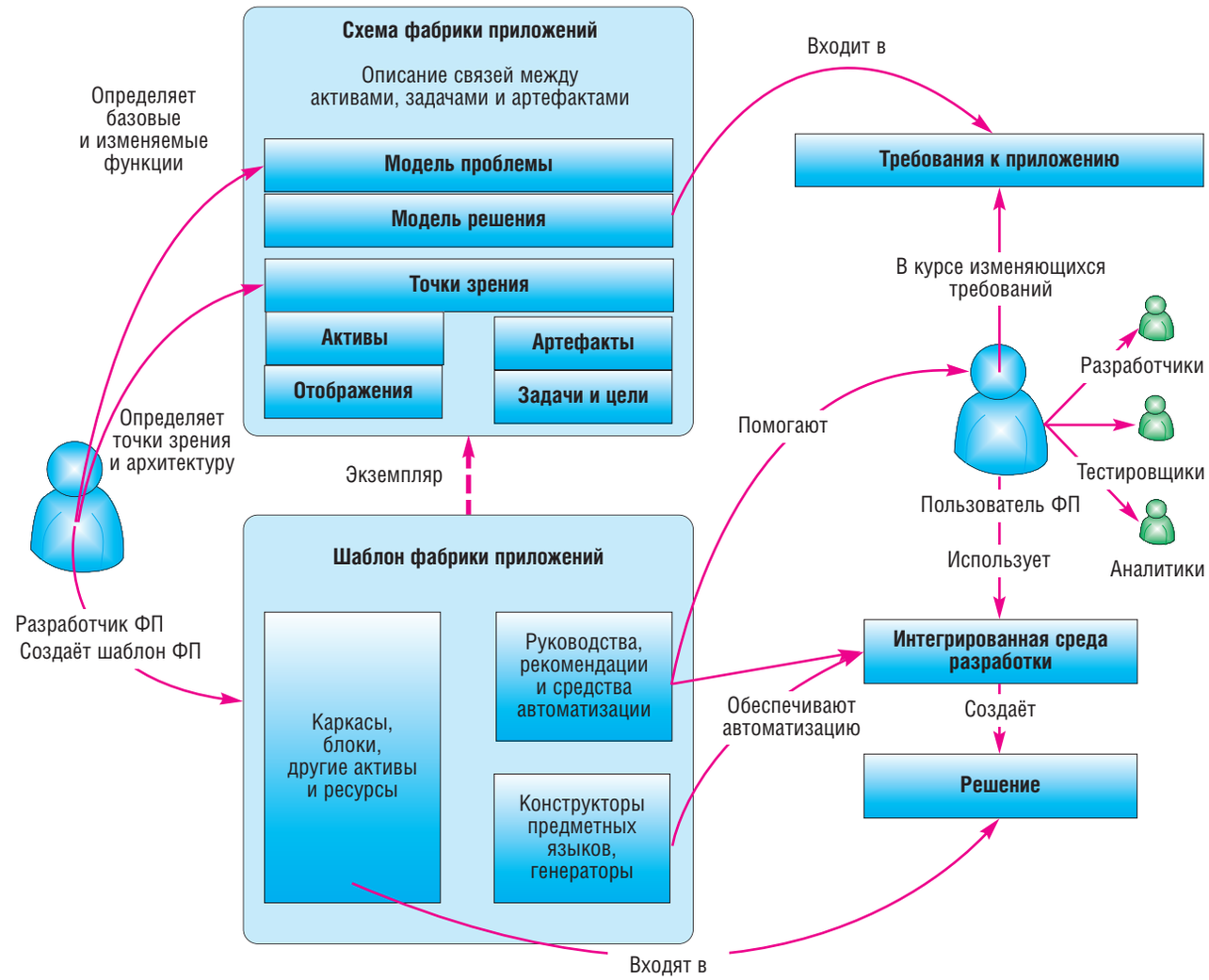

Рис. 2. Фабрика приложений (ФП)

соответствовать требованиям в рамках специфицированного процесса разработки.

Вся фабрика состоит из двух элементов. Это: схема фабрики, играющая главенствующую роль, и шаблон фабрики.

«Схема ФП – модель, описывающая рабочие продукты в заданной предметной области, а также процессы и активы, используемые для их получения» [4].

Схема, описываемая в терминах рабочих продуктов (определенный тип актива), указывает, какие строительные элементы мы используем в ходе разработки продукта. Рабочим продуктом может быть файл, часть файла, несколько файлов или части нескольких файлов. Например, с точки зрения «Библиотеки доступа к данным», рабочий продукт может быть файлом, содержащим класс доступа к данным таблицы базы данных. Вид, определяемый точкой зрения «Библиотеки доступа к данным», может быть проектом, содержащим ряд классов доступа к данным для всех таблиц данной базы данных.

Любая схема ФП состоит из набора взаимосвязанных точек зрения, каждая из которых позволяет взглянуть на систему с разных сторон. Точки зрения помогают понять логическую и физическую сторону организации системы, набор используемых компонентов, каналы дистрибуции и др. Схема фабрика задает отношения между точками зрения.

Точки зрения наглядно документируют архитектуру семейства производимых продуктов. Они могут быть вложенными и определены для разнообразных применений, например, описания различных частей продукта на разных уровнях абстракции, на разных этапах жизненного цикла разработки ПО. Точки зрения могут быть перекрывающимися и модульными. Например, рабочими продуктами для точки зрения «Безопасность данных» могут быть преамбулы методов вставки, замены и обновления классов доступа к данным. Точки зрения чаще всего управляют созданием типов и средств проекта.

Каждая точка зрения должна иметь имя и описание. Необходимо знать производимые или потребляемые точкой зрения рабочие продукты, шаги создания или изменения рабочих продуктов и ресурсы, используемые для выполнения этих шагов. Точка зрения должна сказать разработчику, что строить, как строить и что использовать для построения (модели, средства, шаблоны и т.п.). Точки зрения – стандартные блоки схемы фабрики – формализуют отлаженные действия, которые создают и изменяют рабочие продукты. В зависимости от реализуемой линейки и её продуктов набор составляющих точек зрения может варьироваться. В общем случае состав точки зрения может включать:

- ✧ название;
- ✧ содержание;
- ✧ языковой или аналитический метод построения вида, описываемого точкой зрения;
- ✧ другая мета-информация о точке зрения (источник, автор, дата, ссылки на другие источники и т.п.).

Рабочие продукты, которые производятся или потребляются из данной точки зрения, составляют вид – экземпляр точки зрения, описывающий разрабатываемый продукт с какой-либо стороны. Одна точка зрения может иметь несколько видов. Причем виды могут использовать различные механизмы (языки, модели), позволяющие документировать различные аспекты производства программного продукта. Вид включает следующие элементы:

- ✧ идентификатор вида и его описание;
- ✧ представление системы в соответствии с заданной точкой зрения (может быть представлено в виде модели или текста);
- ✧ конфигурация.

Здесь прослеживается косвенная аналогия между объектно-ориентированным программированием и фабрикой, т.е. отношение между классом и объектом соответствуют отношению вида и точки зрения. Виды позволяют документировать архитектуру ПО, основываясь на точке зрения.

Пример нескольких вложенных точек зрения для некоторого прикладного приложения:

- ✧ бизнес;
- ✧ требования к продукту;
- ✧ архитектура;
- ✧ логическая;
- ✧ распределенная система;
- ✧ компонентная структура;
- ✧ реализация;
- ✧ контроль исполнения;
- ✧ представление бизнес-сущностей;
- ✧ представление данных;
- ✧ структура решения;
- ✧ тестирование;
- ✧ модульное тестирование;
- ✧ интеграционное тестирование;
- ✧ системное тестирование.

Традиционным подходом по выбору точки зрения является использование таблицы, каждый столбец которой соответствует области, а строка – уровню

абстракции *(таб.1)*. Каждая ячейка, стоящая на пересечении строк и столбцов, задает точку зрения.

Таблица 1

Выбор точки зрения

Уровень абстракции	Область			
	Бизнес	Информация	Приложение	Технология
Концептуальный	Диаграммы вариантов использования и текстовые сценарии Бизнес цели и задачи	Бизнес сущности и отношения	Бизнес процессы	Распространение сервисов Качество сервиса
Логический	Определение ролей Модели документооборота	Спецификации документов и сообщений	Взаимодействие между сервисами Определение сервисов Объектные модели	Сервис-маппинг Логика организации сервисов
Физический	Спецификация процесса	Схемы баз данных Стратегия осуществления доступа к данным	Уточненное проектирование Платформо-независимое проектирование	Аренда серверов Установка программного обеспечения Сети

Сам подход, состоящий в применении данной таблицы, не является инновационным. Инновация состоит в том, что он накладывается на семейство продуктов. Случай использования двумерного определения точек зрения имеет ряд недостатков, ограничивающих использование полноценной многомерной схемы фабрики [11].

Схема фабрики – это набор взаимосвязанных точек зрения, необходимых для построения её продуктов. Эту схему можно рассматривать как оглавление, помогающее определить её организацию, чтобы в дальнейшем можно было использовать ресурсы для построения соответствующих продуктов. Схема фабрики во многом подобна схеме *базы данных* (БД), помогающая определить, как организована БД, чтобы можно было обращаться, запрашивать и манипулировать данными, которые она содержит. Схема описывает организацию фабрики; она состоит из точек зрения, связанными с конкретными ресурсами.

В качестве небольшого примера схемы рассмотрим точки зрения фабрики, которая создает приложения для управления предприятием на основе сервис-ориентированной архитектуры [12].

Клиентские приложения. Описывают создание web-приложений или приложений Windows, поддерживающих ввод данных на компьютерах пользователей. Точка зрения сообщает пользователю, как создавать web-формы или формы Windows с помощью конструктора форм и элементов управления для ввода данных из предоставленной разработчиком библиотеки. Действия этой точки зрения направлены на создание web-форм или форм Windows. Рабочие продукты – формы. Ресурсы – конструктор форм и библиотека элементов управления для ввода данных.

Службы процессов. Описывают создание web-служб, ответственных за управление бизнес-процессами. Web-службы всегда создаются единообразно из модели и содержат договор на обслуживание, описываемый C#-интерфейсом с использованием типовых объектов, генерируемых из XSD-файла. Действие для этой точки зрения – создание web-службы. Web-службы – это рабочие продукты. Ресурсы – шаблон и генератор объектов определенного типа.

Службы платформ. Описывают создание web-служб, ответственных только за службы, общие для всех систем, такие, как печать, аудит, авторизация и т.д. Эта точка зрения обеспечивает общие службы, доступные для многократного использования, и сообщает пользователю, как оценить и настроить каждую службу. Действия для этой точки зрения – оценка и настройка служб. Рабочие продукты – настроенные службы. Ресурсы – общие службы.

Перейдем к описанию шаблона ФП. Шаблон – это устанавливаемый пакет с ресурсами, которые имеются в составе фабрики. Без своего шаблона фабрика не будет функционировать корректно.

Шаблон фабрики – экземпляр схемы фабрики, равно как любая модель есть отражение метамодели; совокупность всех активов, определяемых точками зрения внутри схемы фабрики. Все активы могут быть разделены на следующие категории:

- ***библиотеки и каркасы*** – многократно используемые программные компоненты, разрабатываемые в ходе проектирования линейки. NET Enterprise Library – является примером такой библиотеки;
- ***рекомендации.*** Форма предписания и руководства, позволяющая разработчикам решений автоматизировать рутинный процесс;
- ***языки предметной области и дизайнеры*** – задают требуемый уровень абстракции при разработке приложения, а также позволяют генерировать код по созданной модели.

Каждый шаблон ФП включает в себя код и метаданные. Следовательно, каждый из создаваемых шаблонов фабрики может быть загружен в интегрированную среду разработки в целях обеспечения автоматизации разработки. Шаблон фабрики позволяет корректно сконфигурировать все используемые командой разработчиков проекта средства, чтобы в конечном итоге сформировать программный продукт с заданными (предопределенными ранее) параметрами. Это традиционный способ применения шаблонов, абсолютно аналогичный той ситуации, когда вы хотите отослать письмо заданного формата и загружаете в вашу почтовую программу заранее созданный шаблон.

Шаблон должен быть адаптирован под конкретное решение производимого вами продукта. При адаптации схемы вы задаете обозначение того, как ваша фабрика соответствует производимому программному решению, в то время как при адаптации шаблона вы задаете те ресурсы, которые лягут в основу этого решения.

4. Разработка продукта

Фабрики обычно разрабатываются снизу вверх. После определения семейства конечных продуктов при построении системы можно начинать определять и описывать рабочие продукты и действия, организовывать их в точки зрения, разрабатывать ресурсы для поддержки действий, сводить точки зрения в схему фабрики и упаковывать ресурсы в шаблон фабрики. Это наиболее общий подход.

Фабрики служат для быстрой разработки и создания продукта. В данном случае под продуктом понимается продукт из числа семейства продуктов, производством которых мы непосредственно занимаемся. Линейки продуктов определяют:

- ✧ построение новых решений путем сборки частичных решений и/или настройки общих;
- ✧ разработку только уникальных функций;
- ✧ корреляцию вариаций в точках зрения с вариациями в артефактах и процессах;
- ✧ уменьшение «ручной сборки под клиента» на 40–80%.

Этапы разработки программного продукта с использованием фабрик.

- ✦ ***Анализ проблемы.*** Здесь следует понять, является ли разрабатываемый продукт частью развернутой нами фабрики. В зависимости от степени совпадения мы можем отстроить его части вне фабрики как частично, так и полностью.
- ✦ ***Специфицирование деталей производства продукта.*** На этом этапе выявляются требования, предъявляемые к производству заказанного продукта, в рамках требований ко всей линейке производимых продуктов. В зависимости от степени отклонения от требований, можно использовать огромное число механизмов.
- ✦ ***Проектирование продукта.*** Все изменения в требованиях накладываются на линейку продуктов. Воспроизводится архитектура построения продукта, которая адаптированная под уже существующий в рамках фабрики процесс разработки.
- ✦ ***Реализация продукта.*** Разработка компонентов системы, тестов и прочее. Для реализации продукта могут быть использованы различные механизмы.
- ✦ ***Развертывание продукта.*** Повторное использование ограничений на развертывание, конфигураций и т.п.
- ✦ ***Тестирование продукта.*** Здесь используются общие тесты, наборы тестов, тестовые сценарии.

Активы разработки продукта, включающие требования, процесс разработки, архитектуру, компоненты, тесты, схему развертывания, уточняются, повторно используются, адаптируются и организуются в зависимости от точек зрения полученных уточнением схемы фабрики.

Разработка продукта организуется и задается посредством поставленного процесса. Сам процесс организуется при развертывании фабрики, подстраивается под конкретный производимый продукт на стадии адаптации схемы фабрики.

Таким образом, схема фабрики есть некоторый процесс-ориентированный каркас. Мы может работать любым удобным для нас образом, согласно нуждам проекта, среде, определенным обстоятельствам, возникающим в ходе разработки. Главное, чтобы ограничения на процессы, заданные в схеме фабрики, всегда проверялись на соответствие реальной ситуации. Например, можно работать традиционным путем, двигаясь от требований к реализациям, синхронизируя соответствие артефактов.

Ключевые факторы успеха, побуждающие разработчиков принять решения, основанные на подходе фабрик программного обеспечения:

- ✧ универсальность и вариативность;
- ✧ возможность оценивать текущую реализацию процесса разработки продукта с точки зрения производительности и качества;

- ✧ возможность обеспечить прозрачную модель, позволяющую формировать оценки качества и производительности;
- ✧ способность планировать одновременно более одного проекта;
- ✧ быстро и экономно разрабатывать многократно используемые ресурсы, которые инкапсулируют знания и облегчают многократное использование, особенно пользовательские средства;
- ✧ определять конкретные домены и реализовать их с помощью пользовательских средств и процессов, вместо попыток применить средства и процессы общего назначения ко всем доменам.

5. Расширение схемы генерации решения посредством агентов

Базовую схему организации ФП предлагается расширить добавлением некоторого числа агентов. Основная идея в том, чтобы существовало некоторое сообщество или объединение агентов, взаимодействующих как со средой, так и с пользователями ФП. На *рис. 3.* представлена предлагаемая многоагентная архитектура ФП. Здесь КП отображена в виде совокупности БД, объединенных в единую сеть. За ними идет промежуточный служебный слой – сообщество агентов архиваторов, взаимодействующих друг с другом с целью корректного извлечения данных из КП. Они взаимодействуют с агентами медиаторами или промежуточными агентами, отвечающими за служебные функции. Это могут быть функции пополнения списка требований конкретного решения их уточнение и оптимизация. Агент-медиатор позволяет агенту-построителю ФП «наблюдать» за состоянием ФП. КП гетерогенна по своей сути и состоит из множества различных ресурсов, объединенных в общем репозитории. Скорость обновления КП может быть чрезвычайно высокой. Поэтому агенты-построители должны всегда перестраивать схему фабрики для поддержания интеграции как с КП, так и с бизнесом в широком смысле. Можно выделить сообщества агентов ассистентов, основной функцией которых служит адаптация и представление информации для пользователя (интерфейс взаимодействия).

Справа отображено еще одно сообщество (на схеме одна большая фигура агента). Агенты данного сообщества ориентированы на пользователей ФП. Они оптимизируют работу с интегрированной средой разработки программного обеспечения, позволяют работать с требованиями, иметь доступ к руководствам, рекомендациям и другим ресурсам. Они в значительной степени адаптируют среду под каждого члена команды. Если вы разработчик и реализовываете определенный функционал, описанный в шаблоне, агент предоставит вам доступ ко всей информации, которая могла бы пригодиться для реализации, тестирования и отладки. Здесь можно рассматривать как высокоуровневые бизнес требования, так и аспекты, необходимые для реализации.

6. Организация работы членов команд

Рассмотрим вопросы оптимизации работы членов команд. Для этого декомпозируем предыдущую схему и выделим область, касающуюся разработчиков, тестировщиков, аналитиков и других членов команд.

Сегодня существуют попытки тесной интеграции средств разработки и поддержки в целях организация программного рабочего пространства для основных членов проектных команд. Однако они далеки от идеала. Должна существовать тесная интеграция «друг-в-друга» таких инструментальных средств, как системы отладки, контроля версий, интегрированных сред разработки и др. Каждое из этих средств должно быть адаптировано под нужды проекта, а не стоять особняком. Необходимо обеспечить возможности коллаборации между членами команд с целью эффективного решения и повышения уровня коммуникации. Особенно это актуально для крупных компаний, в которых распределённые команды – не редкость.

Рассмотрим, как агенты позволяют организовать такую интеграцию. На *рис. 4* показана прослойка агентов-ассистентов или пользовательских агентов, которые расположены между уровнем интегрированной среды разработки программного обеспечения и самой командой. Агенты, принимая во внимание наличие ресурсов, оповещения от агентов более высоких уровней, помогают пользователям в процессе всего жизненного цикла разработки. Они могут отвечать за лучшие практики в области разработки кода, оптимизируя его и сообщая, что разработчик действует неправильно. На уровне кодирования агенты осуществляют контроль качества. Также агенты обеспечивают актуальность различных моделей. К ним могут относиться различные модели и схемы БД, высокоуровневые диаграммы состояний, модели сценариев, диаграммы классов и др. Следовательно, все, начиная от архитектора, заканчивая разработчиком, видят общую картину.

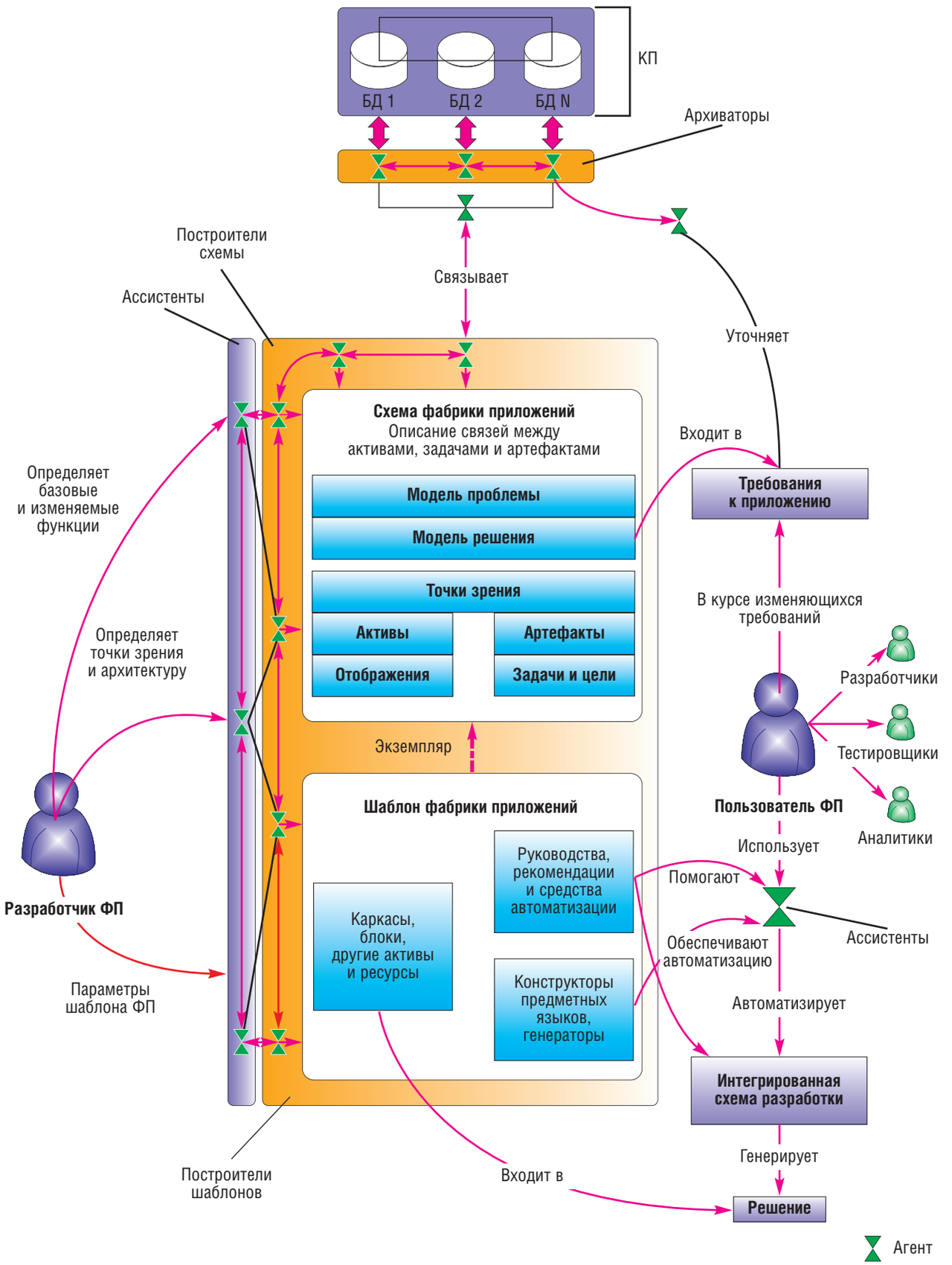

Рис. 3. Многоагентная архитектура фабрики приложений

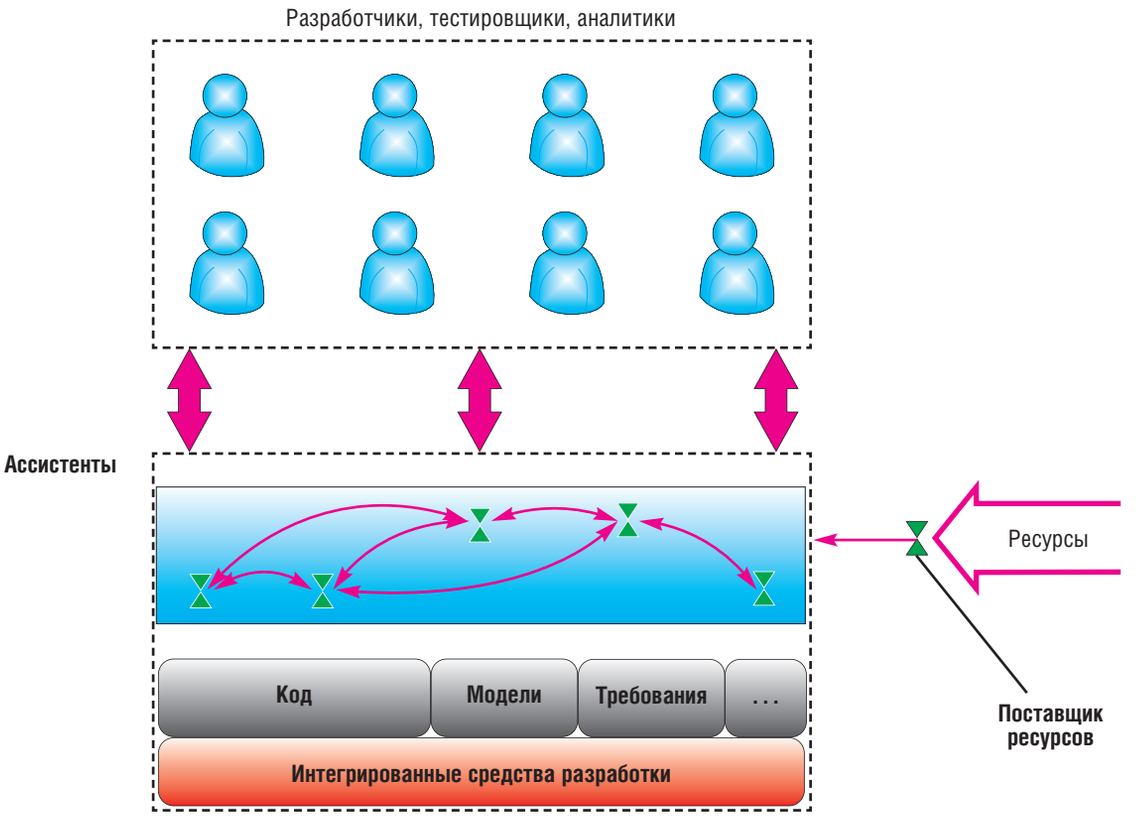

Рис. 4. Взаимодействие проектной команды с интегрированной средой разработки

В качестве инструментальных средств могут использоваться продукты различных компаний. Задача объединить их воедино, максимально интегрировав друг с другом. Это задача агентов. Их сообщество выявляет интересы пользователей, обеспечивая их более персонализированными данными.

7. Общая схема корпоративной агентной ФП

На *рис. 5* представлен вариант преобразования ФП в расчете на развертывание в крупных компаниях. Общая архитектура агентной ФП – трехуровневая. В ней существуют уровень доступа к данным, уровень бизнес логики и уровень представления (интерфейсный или пользовательский).

На самом верхнем уровне архитектуры фигурируют основные источники систематизированных данных компании, а именно, различные КИС. Примерами могут послужить системы управления ресурсами, персоналом, клиентами, системы учета и планирования и др. В случае открытых прикладных программных интерфейсов такие системы предоставляют достаточно средств, для того чтобы сделать их функционирование прозрачным, а извлечение данных эффективным.

Все данные хранятся в КП, «окружённой» агентами-архиваторами, которые работают с ней и предоставляют сервис медиаторам. В свою очередь медиатор нижнего уровня делает прозрачным для агентов уровня логики данных все знания, хранимые в КП. Агенты уровня логики данных «занимаются» анализом данных, в частности в их компетенции задачи управления знаниями, поиска, адаптации данных и обучения. Агенты-построители уровня бизнес-логики отвечают за построение\пополнение схемы фабрики, статистики, учета данных, и прочие активности. Они имеют внешние интерфейсы для взаимодействия с пользовательским слоем, агенты которого связаны с агентами-помощниками, размещаемыми на стороне участников проектных команд.

Таким образом, на основе агентно-ориентированного подхода сформирована высокоуровневая архитектура ФП в рамках предприятия.

Сокращения:

- DSL – domain-specific language – язык предметной области.
- XML – eXtensible Markup Language – расширяемый язык разметки.

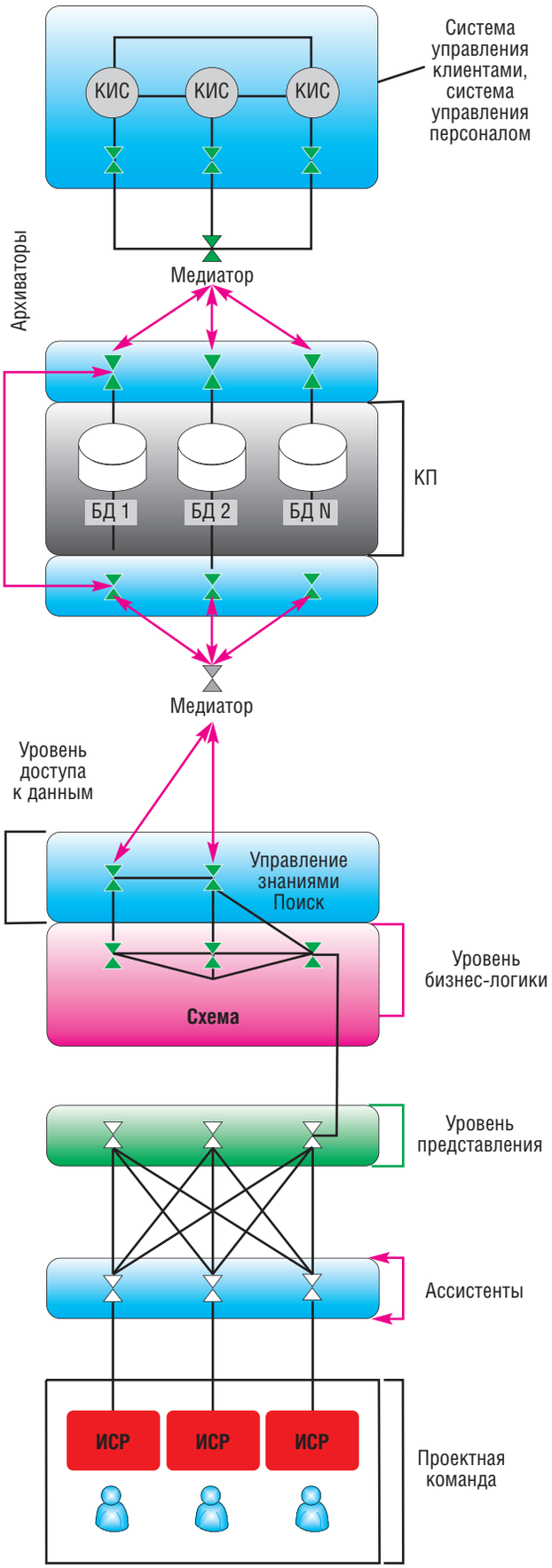

Рис. 5. Агентно-ориентированная фабрика приложений (ФП)

- XSD – XML schema definition (описание структуры документа на языке XML Schema).
- БД – база данных.
- ИСР – интегрированная среда разработки.
- КИС – корпоративная информационная система.
- КП – корпоративная память.
- МАС – многоагентная система.
- СУКП – система управление корпоративной памятью.
- ФП – фабрика приложений. ■

Список литературы

1. Model-driven architecture, http://www.omg.org/mda/
2. Brooks, F. P. Jr., No Silver Bullet. Essence and Accidents of Software Engineering: Computer 20, 1987
3. Brown, Alan W.Component-Based Software Engineering: Selected Papers from the Software Engineering Institute: Computer Society Press, 1996
4. Jack Greenfield, Keith Short, Software Factories. Assembling Applications with Patterns, Models, Frameworks, and Tools: John Wiley & Sons, 2004
5. Berners-Lee T., Hendler J., Lassila, O., The Semantic Web, Scientific American, p. 35-43, May 2001.
6. Wooldridge et al., 2000. Wooldridge M., Jennings N., Kinny D., The Gaia Methodology for Agent-Oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems
7. M. Klush, Intelligent Information Agent: Agent-Based Information Discovery and Managementon the Internet, Springer, pages IX–XVII, 1999
8. Jan Bosch. Design and Use of Software Architecture—Adopting and Evolving a product line Approach. Boston, MA: Addison Wesley, 2000
9. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Reading, MA: Addison Wesley, 1995
10. Microsoft Enterprise Framework & Tools Group – Domain Specific Languages Toolkit, http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx
11. Jack Greenfield, Mauro Regio. Designing and Implementing an HL7 Software Factory: Microsoft Corp., 2005, http://msdn2.microsoft.com/en-us/library/ms954602.aspx
12. Marcel de Vries: Measuring Success with Software Factories and Visual Studio Team System, http://msdn2.microsoft.com/en-us/library/Aa925157.aspx