TechnologiCS 6
процессный подход к разработке и внедрению

Трудности внедрения

Наряду со всеми положительными сторонами комплексного внедрения систем технической подготовки и управления производством всегда существует и ряд объективных трудностей, возникающих при адаптации решений подобного рода. Под объективными трудностями будем понимать такие, которые не зависят от специфики конкретного предприятия или вида и объема внедряемого решения. Такие трудности возникают и у разработчика программного обеспечения, и у промышленных предприятий, которые приступают к комплексному решению своих проблем, опираясь, как правило, только на свой собственный опыт.

Основными трудностями при внедрении комплексных решений для промышленных предприятий являются как большая номенклатура используемых программных средств, связанная с недостаточной функциональностью какого-то одного выбранного продукта или продуктов одной компании-производителя, так и существенно устаревшая нормативная база для решения задач технической подготовки и управления производством, ориентированная на бумажный документооборот.

С другой стороны, разработчика программных средств объективные трудности приводят:

- к необходимости постоянно наращивать функциональность своих продуктов;
- к проблемам с использованием имеющейся функциональности, поскольку на предприятиях, как правило, уже работают решения с похожей функциональностью, но от других производителей – в том числе продукты, разработанные самим предприятием, обычно ориентированные на его специфику.

Объективные трудности внедрения зарубежных продуктов помимо этого связаны не только с адаптацией программных средств к специфике отечественных стандартов, но и с существенными различиями в подходах к подготовке и методах управления производством.

Поэтому если на предприятии полностью отсутствуют программные средства (отметим, что такого практически не бывает) или функциональность имеющихся средств полностью перекрывается системой TechnologiCS (например, задачи конструкторско-технологической подготовки производства (КТПП), см. рис. 1), то, как правило, проблем не возникает и внедрение представляет собой создание новых или конвертацию имеющихся баз данных, а также настройку системы на специфику предприятия.

Совсем другая картина складывается там, где имеющиеся программные средства выходят за рамки функциональности TechnologiCS или попутно с основными решают какие-то специфические задачи, учитывающие особенности конкретного предприятия (например, PDM и ERP, см. рис. 1). Здесь внедрение сталкивается с существенными трудностями, связанными с кропотливой разработкой различного рода регламентов, программных интерфейсов или специальных структур для конвертации и передачи данных из внешних систем и обратно.

Преодоление этих объективных трудностей видится в создании информационных систем, использующих процессный подход и позволяющих как развивать собственные решения, так и использовать решения других производителей программного обеспечения.

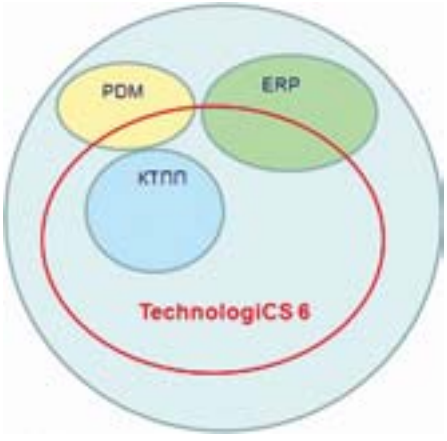

Рис. 1. Трудности решения задач автоматизации

На сегодняшний день большинство имеющихся на рынке систем построено по структурному принципу, то есть имеет свои заложенные при проектировании структуры данных, API для работы с ними и интерфейс пользователя, предназначенный для реализации базовой функциональности системы. Разработчики, занимающиеся проблемами интеграции подобных программных средств, идут, как правило, тремя основными путями:

- используют при построении собственной системы различного рода стандарты для обмена данными между программными средствами, такие как ODMA, группа стандартов STEP (ISO 10303), IGES и т.д.;
- создают программный интерфейс (API), специализированный для решения конкретного круга задач и обеспечивающий функциональность собственной системы;
- разрабатывают специальные структуры данных для обмена между собственными программными средствами, а также конверторы для передачи данных из внешних систем и обратно.

Каждый из этих путей имеет и преимущества, и определенные недостатки. Однако отметим, что для систем, построенных на основе структурного подхода, эти пути являются практически единственными, а самое главное – при их реализации обязательно участие разработчика.

Современные тенденции развития программных средств показывают, что одним из лучших здесь является процессный подход, когда задачи автоматизации формализуются в виде процессов, представляющих собой, как сказано в стандартах на системы менеджмента качества ISO 9000:2000 (п. 3.4.1), "совокупность взаимосвязанных или взаимодействующих видов деятельности, которые преобразуют входы в выходы". Там же дается пояснение, что "любая деятельность или совокупность видов деятельности, которая использует ресурсы для преобразования входов в выходы, может рассма-

триваться как процесс". Других определений процесса, закрепленных на международном или государственном уровне, пока, к сожалению, не существует. Будем также понимать, что процессы состоят из процедур, представляющих собой повторяющиеся последовательности действий, приводящих к определенному результату.

Таким образом, основные задачи автоматизации можно представить в виде набора "кубиков" (процессов), у которых информационные входы одних связаны с информационными выходами других (рис. 2).

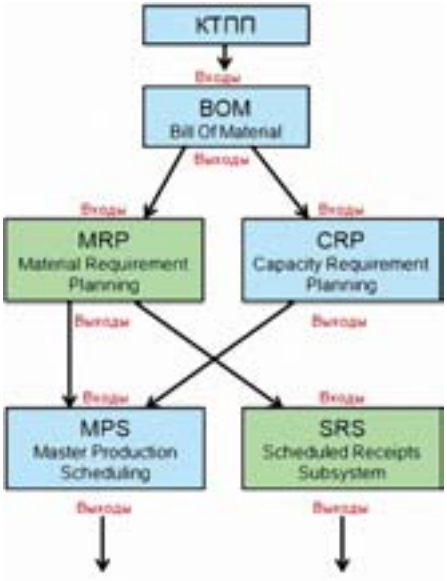

Рис. 2. Взаимодействие процессов, требующих автоматизации

Так, например, систему ERP обычно представляют набором, состоящим из следующих процессов:

- Sales and Operation Planning (SOP) – Планирование продаж и операций;
- Master Production Scheduling (MPS) – Объемно-календарное планирование;
- Inventory Transaction Subsystem (ITS) – Управление движением запасов;
- Scheduled Receipts Subsystem (SRS) – Управление плановыми поставками;
- Material Requirement Planning (MRP) – Планирование потребности в материалах;
- Bill of Materials (BOM) – Ведомость основных материалов;
- Capacity Requirement Planning (CRP) – Планирование потребности в мощностях и т.д.

В условиях, когда заказчик свои процессы уже разработал, утвердил и запустил в промышленную эксплуатацию, система, спроектированная с учетом процессного подхода, должна обеспечивать гибкую настройку на входы и выходы

Рис. 3. Процессный подход к автоматизации

Пожалуй, единственный выход из этой ситуации – предоставить пользователю открытый механизм для построения собственных структур данных. Причем в целях обеспечения целостности данных и реализации прав доступа это должен быть не механизм прямого доступа к таблицам SQL-сервера, а механизм доступа к данным с помощью самих объектов системы. В TechnologiCS 6 такой механизм называется *Наборы данных* (рис. 4) и реализуется с помощью специального инструмента "Визуальный построитель запросов" (рис. 5).

Визуальный построитель запросов предназначен для построения пользователем запросов к объектам TechnologiCS в удобном и понятном виде, как это принято в большинстве информационных систем. Окно построителя запросов состоит из следующих областей:

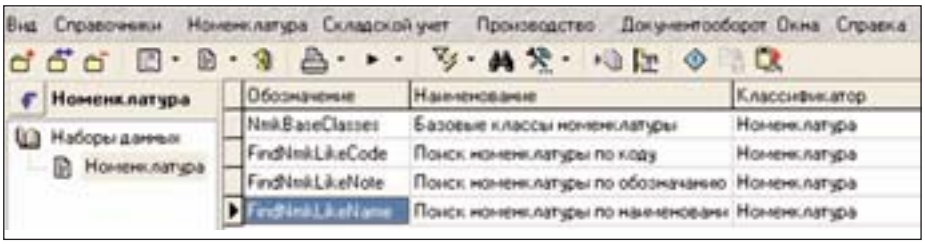

Рис. 4. Справочник наборов данных TechnologiCS 6

имеющихся процессов, которые она либо не покрывает функционально, либо по каким-то причинам принято решение поддерживать ряд процессов уже имеющимся на предприятии программным обеспечением (рис. 3).

Процессный подход к разработке

Любая программная система имеет свои собственные структуры данных. Естественно, эти структуры организованы так, чтобы обеспечить эффективную работу базового функционала системы и, в свою очередь, оптимизированы под имеющийся в системе набор базовых интерфейсов пользователя.

Во время внедрения необходима поддержка процессов, функциональность и объем которых, как правило, заранее неизвестны, и зачастую возникает ситуация, когда для автоматизации процесса необходимо выполнение обработки данных, не заложенное разработчиком в базовую функциональность системы. Построение системы, выполняющей запросы и обработку данных "в обход" заложенной разработчиком структуры, всегда будет стоить очень дорого как с точки зрения машинных ресурсов, так и трудоемкости разработки дополнительного программного кода.

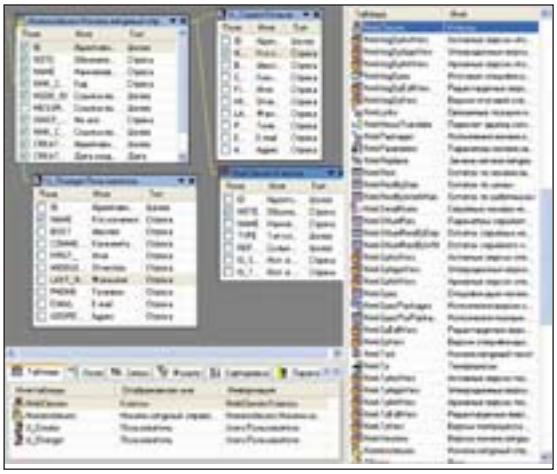

Рис. 5. Визуальный построитель запросов TechnologiCS 6

- **Рабочая область** – область, где отображены таблицы и связи между ними.
- **Область таблиц** – здесь расположен список таблиц, доступных для выбора.
- **Область редактирования** – область, позволяющая изменять набор данных и содержащая следующие закладки:
  - **Таблицы** – содержит список рабочих таблиц;
  - **Поля** – содержит список выбранных для запроса полей;
  - **Связи** – содержит список созданных связей между таблицами;
  - **Фильтр** – настройки фильтра;

- **Сортировка** – настройки порядка сортировки;
- **Расч. поля** – позволяет использовать расчетные и агрегирующие функции в построителе запросов;
- **Макрос** – макрос, который будет использоваться для получения значений расчетных полей;
- **Группировка** – группировка полей для использования агрегирующих функций в расчетных полях;
- **Параметры запроса** – содержит список параметров набора данных;
- **Подзапросы** – содержит список используемых подзапросов (наборов данных) и условия связи;
- **SQL** – содержит текст SQL-запроса;
- **Результаты** – отображает результат запроса в виде таблицы.

Конечно, SQL-запрос виртуален и состоит из объектов TechnologiCS. Тем не менее, пользователь всегда может создавать собственные наборы данных и использовать их в визуальных формах, скриптах, отчетах и т.д., расширяя тем самым базовую функциональность системы.

Наряду с собственными структурами данных любая программная система содержит также набор окон и встроенных алгоритмов обработки данных, которые вместе образуют интерфейс пользователя. Естественно, этот интерфейс является для системы базовым и оптимизирован для работы со встроенными в систему структурами данных. Как правило, степень детализации процессов в таком интерфейсе максимальная, да и встроенные функции сгруппированы не всегда так, как это требуется конечному пользователю. Отсюда и возникают трудности, связанные с чрезмерно перегруженным интерфейсом и необходимостью открывать большое количество окон для реализации, казалось бы, самых простых функций.

Для преодоления таких трудностей в TechnologiCS 6 разработан специальный механизм, позволяющий пользователю создавать для работы собственные экранные формы, задавая им с помощью TechnologiCS API собственные алгоритмы обработки данных. Этот механизм называется "Редактор форм" и состоит из Редактора кода (рис. 6) и Дизайнера форм (рис. 7).

Редактор кода представляет собой среду разработки, предназначенную для создания и редактирования макросов на языке VBScript, и является полнофункциональным редактором текста.

При работе с редактором доступны операции с блоками текста, функции поиска и замены, цветовое выделение синтаксических элементов программных модулей и т.д.

Дизайнер форм, в свою очередь, предназначен для визуального проектирования элементов формы. Таким образом, пользователь может не только самостоятельно разрабатывать собственные структуры данных, но и создавать свои формы для ввода и отображения информации.

Для модификации стандартного и создания собственного интерфейса пользователя существует механизм, называемый Дизайнером интерфейсов (рис. 8).

Этот механизм позволяет сделать систему более гибкой, предоставляя пользователю возможность настраивать конкретные рабочие места, значительно упрощать интерфейс, заданный разработчиком, создавать свои кнопки, экранные формы и макросы.

Таким образом, пользователь может самостоятельно разрабатывать и поддерживать собственные процессы, формируя автоматизированные рабочие места различной функциональности и, что очень важно, – не прибегая при этом к услугам разработчика.

Процессный подход к внедрению

Учитывая имеющуюся функциональность, в процессе внедрения TechnologiCS 6 стал возможен охват более широкого круга задач и выход на качественно новый уровень интеграции с внешними системами.

Во-первых, появилась возможность гибко встраивать систему в имеющуюся на предприятии информационную среду, выполняя последовательно несколько основных шагов:

- Создать описания процессов "как есть". Этот пункт, как правило, является отправной точкой любого внедрения. Здесь очень важно тщательно разграничить функциональные рамки процессов, а самое главное – определить наборы входной и выходной информации для каждого процесса.
- Провести реинжиниринг процессов с указанием того, "как нужно", если в этом есть необходимость.
- Разработать структуры данных для обеспечения информацией входов и выходов процессов.
- Создать структуры с помощью API и инструментов *Наборы данных* и *Визуальный построитель запросов*.

Во-вторых, обеспечить пользователю возможность вводить и обрабатывать

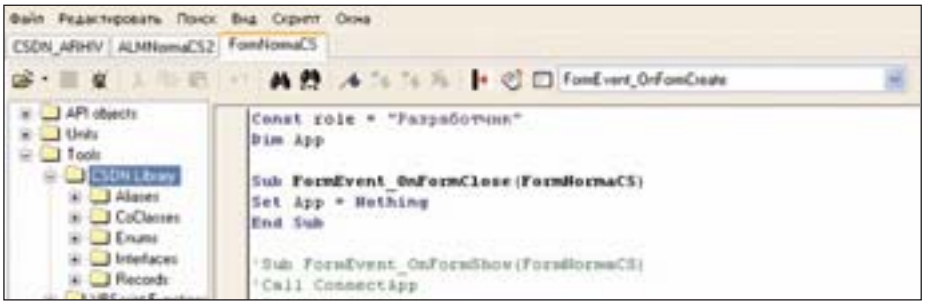

Рис. 6. Редактор кода TechnologiCS 6

Рис. 7. Дизайнер форм TechnologiCS 6

Рис. 8. Дизайнер интерфейсов TechnologiCS 6

данные так, как это ему нужно, даже если такая обработка не входит в базовую функциональность системы. Для этого необходимо:

- разработать и детально описать процедуры, входящие в автоматизируемые процессы;
- с помощью Визуального построителя запросов, API и Редактора форм создать обработчики и экранные формы "как требуется" (рис. 9);
- создать собственные АРМы, поддерживающие разработанные процедуры путем трансформации базовых интерфейсов и/или разработки новых с помощью API и Дизайнера интерфейсов.

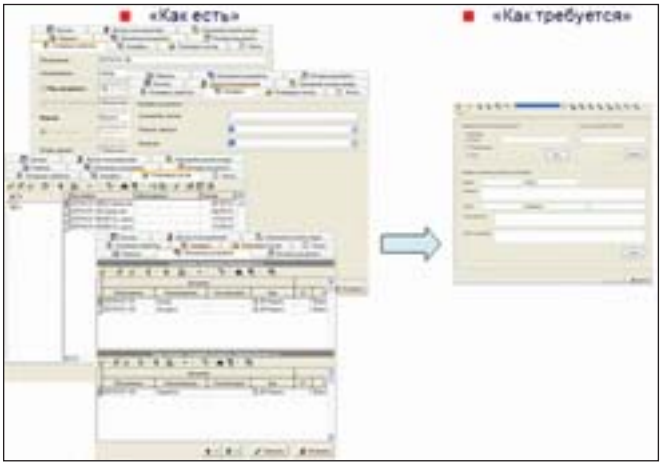

Рис. 9. Построение интерфейсов пользователя "как требуется"

Неоспоримые преимущества

Представленный подход к разработке и внедрению дает системе, спроектированной в рамках процессного подхода, неоспоримые преимущества по отношению к имеющимся на рынке системам, построенным по структурному принципу.

- Пользователь может разрабатывать собственные структуры данных, отсутствующие в системе.
- Поскольку эти структуры основаны не на таблицах базы данных, а на объектах системы, они будут поддерживаться и в будущем, как бы разработчик ни трансформировал внутреннюю структуру данных.
- Разрабатывая собственные структуры, пользователь, тем не менее, остается в рамках единой базы данных, а следовательно, система поддерживает и в будущем всегда будет автоматически поддерживать целостность его данных.
- Помимо обеспечения целостности пользовательских данных, система автоматически разграничивает права доступа к ним в соответствии с общей настройкой системы.
- Пользователь может разрабатывать неограниченное количество АРМов требуемой функциональности, оставаясь в рамках единой базы данных и используя необходимые ему функции базового интерфейса.
- В будущем, в случае изменения состава процедур или объема автоматизируемых процессов, пользователь всегда сможет адаптировать свои структуры данных и разработанные им интерфейсы, не прибегая к услугам разработчика.

Проходит время, и все мы видим, как исчезают порой даже крупные компании и на их место приходят новые разработчики-интеграторы решений. Но независимо от этого, системы, спроектированные в рамках процессного подхода, обеспечивают и всегда будут обеспечивать пользователю возможность самостоятельной поддержки, развития, а следовательно, и существенного повышения устойчивости бизнеса.

Андрей Синельников
CSoft Development Новосибирск
Тел.: (383) 346-0633
E-mail: a.sinelnikov@nsk.csoft.ru

Océ ColorWave™ 600

Высокотехнологичный и экономично выгодный широкоформатный цветной принтер формата **A0**

Нет запаха, эмиссии озона, загрязнения от тонера и чернил, загрязнения окружающей среды

océ
www.oce.ru

Официальный поставщик:
www.csoft.ru
(495) 913-22-22