Сравнение Change Pump, DECS и LEI

Денис Иванов, Владимир Панов, Вадим Федоров

На первый взгляд, технологии DECS/LEI и Change Pump нацелены на одинаковые задачи. Если это так, то ввиду явной невозможности прямо конкурировать с технологиями Lotus и IBM, возникает естественный вопрос: "А нужна ли Change Pump?". Более пристальный взгляд открывает настолько существенные различия в целях, подходах и, следовательно, в возможностях Change Pump и DECS/LEI, что отсутствие прямой конкуренции становится очевидным. Более того, оба подхода дополняют друг друга.

Технологии Change Pump, LEI/DECS организуют взаимодействие компонент системы (блоков), основанное на модели событий. В этой модели какая-либо активность (внутренняя или внешняя по отношению к блокам) вызывает ответные действия в других блоках, причем вид активности и ответные действия задаются *вне блоков*. В следующей таблице показана зависимость важнейших характеристик прикладных систем от выбранной технологии интеграции: Change Pump или LEI/DECS. Последние две технологии объединены, поскольку, их различия сводятся к набору активностей[1].

Таблица 1. Сравнение реализации основных аспектов интеграции в Change Pump, DECS и LEI

Черта	Change Pump	LEI/DECS	Влияние на систему	Вывод[2]
Управление в системе	На основе событий (система реагирует на изменения документов Notes).	В DECS – как в Change Pump, в LEI добавляются другие активности.	Все преимущества и недостатки EDP[3] переносятся с уровня отдельной программы на уровень информационной системы.	
Виды активности	Все виды изменений, отлавливаемые DECS. Создание ссылок. Тиражирование. (Активность "расписание" можно эмулировать с помощью ресурса "Календарь".)	Создание, корректировка и удаление документа Notes. В LEI – еще и копирование по расписанию. (Q1)		
Блокирование интерфейса	Минимально, причем не зависит от сложности	Похоже, что на все время обработки (в	Определяет «реактивность» системы,	●!

[1] DECS - это подмножество LEI, позволяющее работать с активностью real time.

[2] Самые важные характеристики отмечены восклицательным знаком, а победителя указывает буква: '●' - Change Pump, 'L' - LEI/DECS. Нет явного победителя, нет и буквы.

[3] EDP - это программирование, управляемое событиями (event driven programming).

Черта	Change Pump	LEI/DECS	Влияние на систему	Вывод[2]
пользователя	обработки.	случае активности real-time).	воспринимаемую пользователем.	
Устойчивость к временному отказу или перегрузке подписчиков	Да, за счет двойной буферизации в очередях.	Нет (в DECS) и определенное количество повторов в LEI (для активностей, отличных от real time). Для активностей, отличных от real-time, **возможен пропуск изменений**.	Это свойство необходимо для отказоустойчивых систем. Технология, не обеспечивающая отказоустойчивости, вообще, не может быть системообразующей. Повторы LEI не обеспечивают надежности, поскольку очередной повтор может попасть на измененные данные.	•!
Динамика связи источника изменений с получателем	Асинхронная связь.	Синхронная связь. В LEI – еще и расписания и программируемые активности.	Асинхронная связь повышает отказоустойчивость системы и готовность интерфейса пользователя. Последний заблокирован лишь на время формирования события и передачи его диспетчеру	•!
Приоритеты в работе с получателями	Динамическое управление приоритетами входных очередей подписчиков	Активностям могут быть присвоены постоянные приоритеты.		•
Соответствие структуры объектов в издателе и подписчиках	И со стороны издателя и со стороны подписчика - весь спектр: от точного соответствия до полной маскировки.	Если пользоваться только метаданными, то – сильная структурная связь. Метаконнекторы и агент требуют программирования. (Q3)	Независимость структур события и сигнала от структур измененного и изменяемого объектов важна при расширении системы и связывании неоднородных систем.	•
Компоновка события из нескольких документов	Нет, но в событие попадают атрибуты подписчика, модули и вычисленные поля, а в сигнал - еще и атрибуты получателя. Метод LEI чужд Change Pump поскольку в других документах нет изменений	Да, для цепочек ответных документов.	Что-то подобное схеме LEI пригодилось бы и в Change Pump. Естественный путь - механизм "мастеров", которые при создании заполняются информацией из других документов, а при завершении работы распространяют	L!

Черта	Change Pump	LEI/DECS	Влияние на систему	Вывод[2]
			изменения по стандартной схеме Change Pump. ("Мастер" напоминает карту транзакции.) Этот механизм позволяет правильно управлять ключами блокировки документов и обнаруживать конфликты тиражирования.	
Информация, передаваемая событием	Вид изменения, исходное и конечное состояния.	Вид изменения и конечное состояние.	Без исходного состояния практически нельзя автоматически поддерживать актуальность таблиц сопряженности и сходных видов сводок.	•
Управление со стороны приложения параметрами связей, заданных метаданными	Функции полей изменяемого документа, модули, атрибуты издателя и подписчика, данные в событиях и сигналах, данные из документа-получателя	Формулы, вычисленные по полям изменяемого документа. (**Q3** и **Q4**)		
Реакция на несогласие подписчика	Откат или исключительная ситуация в соответствии с метаданными. Механизм исключительных ситуаций позволяет локализовать пораженную область и защитить ее от новых воздействий.	Ошибка при обработке может вызвать повторную обработку.	Откат изменений особенно важен при взаимодействии приложений Notes с "чужеродными" системами (например, РСУБД). Как администратор справится с ошибками без механизма исключительных ситуаций?	•!
Защита критических функций от внешних помех	Блокировка сервера/базы данных/документа на уровне хука.	Нет.	Без блокировки невозможно обеспечить целостность транзакций.	•!
Равноправность «чужеродных» частей системы (например, реляционных БД)	Активны т.к. все ИР (включая шлюзы) и полностью равноправны	Пассивная реакция (активность real time) на изменения в «ведущей» БД Notes. В LEI – polling и программное управление (**Q?**)		•!

Черта	Change Pump	LEI/DECS	Влияние на систему	Вывод[2]
Функциональные возможности шлюзов в «чужеродные» системы	Интеллектуальные шлюзы любой сложности	Скорее всего, - примитивные: преобразование форматов данных, инкапсуляция API и протоколов взаимодействия с чужеродными системами. Ясности не будет до выхода SDK).	Внутри шлюза сложная обработка нужна для переупорядочивания событий и др. Зато подход DECS повышает эффективность и позволяет связи работать в режиме реального времени. С существующими коннекторами LEI можно работать из Java, используя их в шлюзах Change Pump.	!
Установка ссылок	Да, с маскировкой структур объекта и субъекта ссылки (вплоть до ссылок между чужеродными объектами).	Нет самого понятия «установка ссылки» (Q3)		.!
Обновление ссылок	Автоматическое (по настройкам подписок и стандартных реакций).	Вероятно, это позволяют сделать процедуры, вызываемые активностями. (Q5)		!
Автоматическое «расширение» ссылки при открытии документа.	Нет, но простота обновления ссылок снижает потребность в этом свойстве.	При открытии документа – динамическая подстановка значений, скопированных из документа, на который направлена ссылка.	Динамическая подстановка значений в некоторых случаях очень полезна, но ее систематическое использование вызывает обилие лишней работы и снижает отказоустойчивость.	!
Создание выборок[4]	Диспетчер распространяет запрос ИР во все ИР, где могут находиться объекты указанного класса (и наследники) и возвращает поток ссылок.	Extended Search (ES)	ES не связан с LEI	.!
Обновление выборок, полученных по запросам	Автоматическое, причем адресата задавать не нужно.	Да (опрос и м.б. Extended Search). Q6		.!

[4] Запросы и выборки пока исключены для экономии времени.

Черта	Change Pump	LEI/DECS	Влияние на систему	Вывод[2]
Обновление таблиц сопряженности и сходных сводок	Автоматическое (по настройкам).	Нет		.!
Сводки, непротиворечивые на определенный момент.	Да, путем временной задержки входной очереди подписчика.	Требует копирования большого объема данных во временное хранилище.		.!
Моделирование предметной области	Метаданные – это модель предметной области, в которой отражены множественные логические отношения «общее-частное» (наследование и адаптеры), сходство поведения (групповые подписки) и ссылочная структура, а особенности представления данных замаскированы.	Нет, описание активности опирается на структурные элементы БД Notes		.!
Нужны ли изменения в дизайне приложений	Без изменения дизайна можно добиться большего, чем в LEI. Для обновления ссылок и зависимой информации нужны минимальные изменения.	Нет (за исключением добавления процедур в получателей)	В реальной системе влияние на дизайн исключительно важно.	.!
Территориально распределенные системы	Прозрачное для издателей и подписчиков взаимодействие диспетчеров через слой Company Media	Локальное решение, но при соответствующих настройках (расписание тиражирования, активности, …) что-то может получиться.		!

Дополнительные замечания об архитектуре DECS и LEI/DECS

Архитектура DECS и LEI напоминает другие инструменты интеграции Domino с "чужеродными" системами, например, Extended Search, MQ Connector и т.п. Для этой архитектуры характерны следующие черты:

- Все взаимодействия – только межсерверные, ПО не работает на сервере Local.
- Для непосредственного взаимодействия с «чужими» серверами используются простые (по набору функций) драйверы. Имеется SDK для разработки на

C/C++ новых драйверов третьими фирмами. Драйверы не опираются на открытые стандарты, например, JDBC.

- Работой сервера интеграции, находящегося между сервером Domino и драйвером, управляют внешние настройки – БД Notes или LS и Java-программы, использующие LC LSX.
- Перенос информации рассматривается как неделимый «мгновенный» процесс и поэтому даже не возникают раздумья о внутренней структуре и функциональности объекта, представляющего передаваемую информации на ее пути между серверами данных. Это усиливает структурные связи между частями системы, реализованных на этих серверах.

Соображения по выбору технологии интеграции

Все рассмотренные технологии интеграции существенно различаются по назначению.

- LEI/DECS действуют на близком расстоянии (связывают компоненты в пределах локальной сети). У Change Pump такого ограничения нет.
- Пакетная передача больших объемов данных, в которой главное – это преобразование их формата, требует LEI.
- Простейшие реакции, которые должны незаметно уместиться в действие (например, подстановка значений вместо кодов), приводит к DECS. На DECS можно реализовать и простую синхронизацию пары справочников, но уже в этой задаче у Change Pump есть преимущества, которые усиливаются с усложнением системы (ссылки, выборки, сводки, полиморфизм подписчиков и др.).
- Желание построить модель предметной области, получить внешне быстрое и устойчивое к отказам приложение приводит к Change Pump.

Выводы

- Технологии Change Pump и DECS/LEI предназначены для решения различных задач, причем эффективно решить целевые задачи одной технологии с помощью другой нельзя.
- Архитектура DECS и LEI кардинально отличается от архитектуры Change Pump, сильные стороны последней (системообразующая модель «издатель-подписчики», асинхронность связей, встраивание в процессы корректировки информации пользователями, репликатором и агентами) нельзя получить в результате косметических изменений архитектуры LEI/DECS.
- Маловероятно, что Lotus/IBM повернется от LEI/DECS в сторону решения, аналогичного Change Pump, поскольку последнее навязывает конкретную архитектуру прикладных систем на Notes, а не является нейтральным к ней механизмом интеграции. В настоящее время IBM/Lotus ориентированы на крупных заказчиков, и позиция IBM состоит в том, чтобы решать задачи Change Pump, комбинируя Domino с помощью LEI/DECS с уже имеющимися технологиями (MQ, DB2, Data Proragator, CICS и т.д.). Пока нет никаких прямых или косвенных признаков того, что в Lotus/IBM зарождается другой продукт, близкий к Change Pump. Вероятно, этого и не будет.
- И в СНГ и за рубежом небольшие фирмы (составляющие в сумме до 70% экономики) испытывают реальную потребность в приложениях Notes,

использующих все эти вещи. Но малым фирмам решение Lotus/IBM не подойдет из-за сложности и стоимости. Кроме того, это решение в большей степени ориентировано на интеграцию крупных инфраструктур, а не на связывание частей прикладной системы на Notes. Это – наша ниша.

- В то же время, игнорировать DECS и LEI нельзя, Change Pump должна быть совместима с ними. Возможные точки соприкосновения:
  - Шлюз Change Pump взаимодействует с внешними хранилищами через DECS (ради коннекторов), а вся сложная обработка происходит в нашей части шлюза.
  - С созданием событий – неясно. Во всяком случае, заменить hook Change Pump на DECS не удастся.
  - Шлюз Change Pump с DECS можно более глубоко интегрировать в Change Pump, чем обычный шлюз. Например, DECS может обеспечивать обработку сигналов при их появлении во входных очередях подписчиков.
  - В обычных шлюзах Change Pump надо использовать коннекторы DECS/LEI.

Вопросы по DECS/LEI

Q1. Как DECS работает с каскадами изменений и открытий?

Q2. Можно ли приоритетами повлиять на логику?

Q3. Можно ли скрыть структуру получателя с помощью метаконектора (Connection Broker)?

Q4. Можно ли программно установить соединение с источником, не описанным в метаданных?

Q5. Можно ли описать программную реакцию (воздействие на БД Notes как получателя) на real time activity?

Q6. Обновление выборки: автоматический повтор запроса.