Открытость библиотечных систем
Openness of Library Systems
Відкритість бібліотечних систем

Баранов В . Л .

Санкт-Петербургский государственный политехнический университет, Санкт-Петербург, Россия

Vladimir L. Baranov

St. Petersburg State Technical University, St. Petersburg, Russia

Баранов В. Л

Санкт-Петербурзький державний політехнічний університет, Санкт-Петербург, Росія

Рассматривается понятие открытости информационных систем применительно к библиотечным системам. Подробно рассматриваются основные свойства открытых систем, раскрывается их содержание, базис и реальная значимость для библиотечных систем. Предлагается рассматривать открытую архитектуру библиотечных систем как концепцию их развития.

The definition of openness of information systems in respect of library systems is analyzed. Major properties of open systems, their content, basis and real value for library systems are considered. The architecture of library systems is proposed to be considered as their development concept.

Досліджуються поняття відкритості інформаційних систем стосовно бібліотечних систем. Детально аналізуються основні властивості відкритих систем, розкривається їх зміст, базис і реальна значимість для бібліотечних систем. Пропонується розглянути відкриту архітектуру бібліотечних систем як концепцію їх розвитку.

Библиотечные системы (БС) являются сложным высокотехнологичным программным обеспечением (ПО), разновидностью информационных систем (ИС). И как любое открытое ПО, открытые библиотечные системы должны удовлетворять свойствам открытости. В [1] приводится определение термина открытой системы, сформулированное специалистами IEEE, которое подчеркивает аспект среды, которую предоставляет открытая система для ее использования. Согласно ему, “открытая система” определяется как “исчерпывающий и согласованный набор международных стандартов информационных технологий и профилей функциональных стандартов, которые специфицируют интерфейсы, службы и поддерживающие их форматы, чтобы обеспечить интероперабельность и мобильность приложений, данных и персонала”. Свойствами открытых систем являются [1]:

расширяемость/масштабируемость: обеспечение возможности добавления новых функций ИС или изменения некоторых уже имеющихся при неизменных остальных функциональных частях ИС;

мобильность/переносимость: обеспечение возможности переноса программ, данных при модернизации или замене аппаратных платформ

ИС и возможности работы с ними специалистов, пользующихся информационными технологиями, без их переподготовки при изменениях ИС;

интероперабельность: способность к взаимодействию с другими ИС;

дружественность к пользователю.

Приведенное выше определение и свойства понятны специалистам, но мало что говорят обычным библиотечным работникам, за исключением разве что “туманного” свойства “дружественность к пользователю”, которое каждый понимает по-своему, не исключая разработчиков. “Туманным” его можно считать, поскольку оно требует слишком развернутого пояснения, и понимается скорее интуитивно. Большинство людей сразу думают о графическом интерфейсе, хотя последний сам по себе не обязан быть “дружественным”.

Таким образом, когда словом “открытость” хотят подчеркнуть какие-то качества конкретной БС, то для большинства конечных пользователей (библиотекарей) это либо “пустой звук”, либо это слово означает “что-то очень хорошее и положительное”. Естественно, что последнее ближе к истине. Даже если иметь в виду пояснение свойств открытости, данное выше, все равно они требуют дополнительного объяснения с указанием конкретных функциональных возможностей. Т. е. слово “открытая” не совсем полно и точно отражает то, что мы хотим выразить этим словом для конечного пользователя. Это понятие для разработчиков ИС, и в частности БС. Конечному пользователю нужна конкретная функциональность, а не теоретические формулировки. Тем более что имеющаяся в ИС “открытость” зачастую не выходит за рамки “теоретической возможности”, т. е. не реализуется практически ввиду отсутствия объектов приложения открытости (скажем, для реализации интероперабельности). Чтобы наполнить понятие “открытая” конкретным содержанием, необходимо перевести теоретические формулировки в практическую плоскость для конкретного класса ИС, а именно, в нашем случае, для БС. Другими словами, открытая БС должна реализовывать совершенно определенный набор функций. Этот набор, естественно, можно дополнительно поделить на уровни.

В данной статье не ставится задача предложить один из вариантов набора функций для открытой БС, а лишь раскрыть содержание, базис и реальную значимость первых трех свойств открытости для БС.

Перед тем как перейти собственно к свойствам, хочется дать несколько замечаний. Идеальной представляется такая ситуация, когда существует **открытая архитектура библиотечных систем**. Под *открытостью* здесь подразумевается в первую очередь “общепринятость”, основанная на стандартах, минимальных требованиях к реализации, рекомендациях и т. п. Если бы все разработчики БС создавали свои системы на базе открытой архитектуры, то проблем с открытостью не было бы. Однако это в целом утопия, поскольку такой архитектуры как целостной системы нет, а БС уже есть и приведение существующих БС к базису открытой архитектуры (которой тоже пока нет), скорее всего, будет коммерчески нецелесообразно. Следует отметить, что такие работы стоят очень дорого, а рынок БС гораздо меньше, чем рынок другого ПО (операционные системы, офисные системы, бизнес системы и пр.), где затраты на развитие быстро окупаются. Несмотря на вышесказанное, создавать открытую архитектуру имеет смысл, но не как некоторую вещь в себе, оторванную от практики, а как **концепцию развития** БС, не навязывающую конкретных сиюминутных решений. Необходимо обеспечить

участие в разработке концепции производителей БС. Как показывает опыт “запада”, для создания и продвижения новых стандартов обычно создаются консорциумы из нескольких производителей. Конечно, достаточно крупная компания (типа Microsoft) может решить подобную задачу в одиночку, но это в случаях, когда компания может диктовать свою волю другим или когда не ставиться задача обеспечения открытости.

Можно выделить следующие основные задачи в рамках концепции развития БС:

Разработка общих спецификаций построения стандартов взаимодействия компонент (модулей) БС, как автономных (имеющих самостоятельное значение), так и зависимых (интегрирующихся). Имеется ввиду не только сетевое взаимодействие, но и обычное межпрограммное.

Обеспечение высокой динамики и гибкости (расширяемости с сохранением совместимости) развития стандартов взаимодействия, сокращение сроков их внедрения. Для этого при разработке или модификации стандартов необходимо учитывать существующее положение вещей, возможности различных БС. Также необходимо, с одной стороны, в максимально возможной мере следовать международным и государственным стандартам, а, с другой стороны, не превращать эти стандарты в идолы, не допускать, чтобы они мешали реальной работе и развитию. В последнем случае лучше отдавать предпочтение отраслевым стандартам, нежели международным, которые, как известно, неповоротливы и опаздывают (если за ними не стоит крупная компания).

Выработка гибкой политики взаимодействия производителей БС, которая обеспечивала бы не только взаимодействие между различными БС как автономными системами, но и программную интегрируемость модулей разных производителей. Например, чтобы, имея некоторую БС, можно было использовать модуль анализа и представления статистики (или, например, модуль книгообеспеченности) от другой системы или такой модуль от производителя, который специализируется именно на таком ПО (не занимаясь разработкой собственно БС). Хорошую БС сделать трудно, но еще труднее сделать такую систему, которая бы превосходила все другие по всей функциональности и ее качеству. И производители, которые разрабатывают не БС целиком, а некоторые ее компоненты, могут добиться лучших результатов. Например, несмотря на развитие систем управления базами данных, продолжает существовать ПО сторонних производителей для генерации отчетов. Кроме того, у разных пользователей разные представления о качестве, удобстве. С этой точки зрения создание идеальной БС практически невозможно.

Рассмотрим теперь свойства открытых систем. Первое свойство: *расширяемость/масштабируемость*. Обычно под этим понимают возможность исполнения ПО в распределенном режиме, т. е. на нескольких процессорах, на нескольких компьютерах, а также возможность наращивания нагрузки при соответствующем наращивании мощности аппаратуры (от тривиальной до создания кластеров), увеличения количества рабочих станций в сетевом аспекте. Однако в настоящее время это хоть и важно, но не так актуально, поскольку обеспечивается существующими технологиями. Актуальным является функциональный аспект, как раз и отраженный в данном выше пояснении рассматриваемого свойства. Что же дает

данное свойство обычному конечному пользователю? Да практически ничего. Однако оно много значит для разработчиков, позволяя снизить затраты и сохранить инвестиции. Также оно много значит для тех конечных пользователей, которые не способны сами разработать БС целиком, но способны модифицировать имеющие функции под свои нужды или добавить новые. При наличии грамотных спецификаций это реально. Однако здесь таится опасность настолько видоизменить систему или отдельные ее функции, что она станет неспособна к дальнейшему развитию. Открытая архитектура позволит избежать этого. Следует также отметить, что особо больших достижений здесь ждать не следует. В крайнем случае, вместо полноценной БС мы получим конструктор “сделай сам”, что большинству пользователей БС не нужно.

Следующее свойство: *мобильность/переносимость*. Это свойство, без сомнения, является важным для конечного пользователя. Однако в том виде, как оно пояснено выше, при существующем уровне программных технологий, оно выполняется практически само собой. И здесь гораздо важнее возможность переноса при замене программной платформы. И здесь мало утешительного. Если при замене операционной системы на следующую версию данной же системы все может пройти более-менее гладко, то при замене другого ПО, на котором основано БС, потребуется значительные изменения и, как следствие, новый виток отладки уже, в общем-то, отлаженных функций. Подобная ситуация произошла при включении в БС “Руслан” поддержки системы управления базами данных Oracle 9i при уже реализованной поддержке Oracle 8i. Чем больше в некотором ПО включений ПО сторонних фирм, тем сложнее ситуация, но от этого никуда не уйти, поскольку это значительно сокращает время разработки и затраты. Вернемся опять к операционным системам. Одни пользователи предпочитают Microsoft Windows, другие — UNIX-подобные системы, третьи — для серверных модулей — UNIX, а для рабочих станций — Microsoft Windows. Если бы ПО было платформо-независимым, это было бы прекрасно. И в принципе это возможно (на уровне исходных кодов конечного ПО). Однако поскольку ПО создается с помощью другого ПО — сред разработки, то и его переносимость зависит от переносимости этих сред или возможности использования аналогичных сред на другой платформе. С другой стороны, при использовании существующего ПО, также требуется его переносимость. С учетом вышесказанного реализовать платформо-независимую БС возможно, но требует значительных усилий, причем с самого начала процесса разработки. В дальнейшем, при неправильном выборе среды разработки или выборе используемого ПО сторонних фирм, потребуется на порядок большие усилия для реализации переносимости. А ресурсы производителей БС, как уже говорилось, ограничены в связи с ограниченностью рынка. Поэтому, например, новая версия БС “Руслан” будет реализована с учетом вышесказанного, но, тем не менее, обеспечение многоплатформенности все равно потребует дополнительных затрат.

И последнее свойство, которое хотелось бы обсудить — это *интероперабельность*. Этому свойству в области БС в последнее время уделялось особое внимание. И данное выше пояснение этого свойства, как нельзя кстати, подходит для области БС. Под взаимодействием обычно понимают сетевое взаимодействие. Однако сюда можно и отнести и опосредованное взаимодействие через внешние файлы данных. И здесь, на основании опыта, полученного в процессе как эксплуатации БС, так и в процессе перехода с одной БС на другую, могу констатировать, что важнейшей функцией, с которой начинается открытость — это возможность экспорта/импорта всех (возможно, за исключением служебной информации) данных из/в БС в формате, стандартном для данной области (в нашем случае это MARC и ISO2709 для библиографических записей) или, для “нестандартных” (не библиографических) данных, в формате с открытом описанием, позволяющем в полной мере отразить структуру данных. Перейдем теперь к

сетевому взаимодействию. В области БС сетевое взаимодействие в основном (по крайней мере, в России) связывают с протоколом Z39. 50, который является международным стандартом. Следует заметить, что этот протокол не предназначался для организации взаимодействия БС. Он предназначен для организации взаимодействия клиентского ПО с серверным ПО, хотя и то и другое могут принадлежать различным БС. При этом основной причиной, сдерживающей его распространение, является его сложность. Сразу можно сказать, что для той функциональности, которую обеспечивает этот протокол, он не является сложным. Другое дело, что большая часть этой функциональности в большинстве случаев не будет востребована на практике, а для того чтобы реализовать собственно межсистемное взаимодействие потребуется реализовывать и клиент и сервер. Например, в случае почтовых серверов, протоколы взаимодействия самих почтовых серверов и почтовых клиентов с сервером различаются. Поэтому под взаимодействием по Z39. 50 в основном понимается возможность клиентского ПО одной БС взаимодействовать с серверным ПО другой БС. При этом взаимодействие внутри самой БС может осуществляться как по протоколу Z39. 50, так и другим способом. Однако, в случае клиент-серверного взаимодействия между разнородными БС, достойных альтернатив протоколу Z39. 50 пока нет. Тем не менее, работы в данном направлении идут. Сам протокол Z39. 50 также развивается, в том числе и в направлении упрощения работы с ним (SRW — Search/Retrieve Web Service, ZOOM — объектно-ориентированный программный интерфейс).

Если поставить задачу организации межсистемного взаимодействия наиболее простым и открытым способом (в рамках рассматриваемой выше концепции развития), то наилучшим решением в ряде случаев (зависит от требуемой функциональности) будет разработка *специализированного прикладного протокола*. Еще несколько лет назад разработка протокола казалась очень сложной задачей. Однако современные технологии программирования позволяют реализовать сетевое взаимодействие (как передачу протокольного блока данных с одного узла сети на другой) на таком уровне, когда оно приближается по сложности к взаимодействию посредством обычных программных интерфейсов (API). С другой стороны появление языка описания структурированной информации XML и соответствующего ПО для работы с ним практически до предела упрощает работу с содержательной частью протокола (протокольными блоками данных). Однако сам по себе XML не создает открытость, это лишь удобное (для решения определенных задач) средство. Более того, в ряде случаев с простой функциональностью можно решить задачу сетевого взаимодействия гораздо проще без использования XML. Но в любом случае для того, чтобы протокол получил широкое распространение, необходимо вести разработку в рамках концепции развития, изложенной выше.

В заключение можно сказать, что открытость БС в большей степени реализована на бумаге, чем на деле. И это не так плохо — поставлены ориентиры развития. Только не следует забывать, что открытость — не самоцель, она должна работать на практике. Другими словами, лучше, если на практике мы получим реально взаимодействующие, интегрирующиеся системы на базе менее открытых, но более работоспособных отраслевых стандартов. На практике может оказаться, что гораздо проще реализовать поддержку международного стандарта в общем (для всех БС) “шлюзе” или модуле, чем внедрять его повсеместно.

Литература

Филинов Е. Выбор и разработка концептуальной модели среды открытых систем. // Открытые системы. — 1995. — №6. (http: //www. osp. ru/os/1995/06/71. htm)