Часть 1. Сценарии и решения использования шины Enterprise Service Bus в сервис-ориентированной архитектуре

Функции Enterprise Service Bus

Рик Робинсон, архитектор информационных систем, IBM

Описание

Автор определяет минимальный набор функций, обеспечивающих выполнение большей части требований к Enterprise Service Bus (ESB) в соответствии с принципами сервис-ориентированной архитектуры (Service-Oriented Architecture, SOA). Определение минимального набора функций позволит разобраться в том, какие из имеющихся технологий следует использовать для реализации ESB, поддерживающей SOA. Поняв, какие дополнительные функции диктуются требованиями конкретной ситуации, вы сможете выбрать наиболее подходящую в этой ситуации технологию реализации.

Введение

Современная интеграция информационных систем представляет собой реализацию сервис-ориентированной архитектуры (Service-Oriented Architectures, SOA) с использованием технологий Web-сервисов; существует много превосходных описаний преимуществ и методов такой реализации (см. раздел Ресурсы). В последнее время ключевым компонентом инфраструктуры SOA считается корпоративная сервисная шина *Enterprise Service Bus* (ESB) (см. раздел Ресурсы). Однако важно точно знать, что такое ESB - продукт, технология, стандарт или что-либо еще. В частности, можно ли построить ESB сегодня, и если да, то как именно?

В этой статье ESB описывается как набор функций инфраструктуры, реализуемый при помощи технологий связующего ПО, обеспечивающих поддержку SOA. ESB поддерживает взаимодействия с использованием сервисов, сообщений и событий в неоднородной среде, с надлежащим уровнем сервиса и управляемостью. В статье мы собрали и классифицировали самые различные функции. Однако не во всех ситуациях работы с ESB обязательно использовать все эти функции.

В статье определяется минимальный набор функций, обеспечивающих выполнение большей части требований к ESB в соответствии с принципами SOA. Определив минимальный набор функций, мы сможем понять, какие из имеющихся технологий можно использовать для реализации ESB, поддерживающей SOA. Поняв, какие дополнительные функции диктуются требованиями конкретной ситуации, вы сможете выбрать наиболее подходящую в этой ситуации технологию реализации.

В следующих статьях будет описан набор ESB-сценариев в обычных исходных точках SOA для реализации ESB или SOA. В свою очередь, шаблоны решений помогают выбрать подходящие технологии для реализации.

Поскольку развиваются ситуации, в которых используется решение ESB, соответствующим образом развиваются и необходимые для ESB функции. Аналогично будут развиваться функции и средства явно использующих ESB продуктов. Поэтому в заключительной статье данной серии мы рассмотрим схему внедрения SOA и ESB, чтобы дать рекомендации на первом этапе использования функций и технологий ESB и продемонстрировать возможности постепенного внедрения.

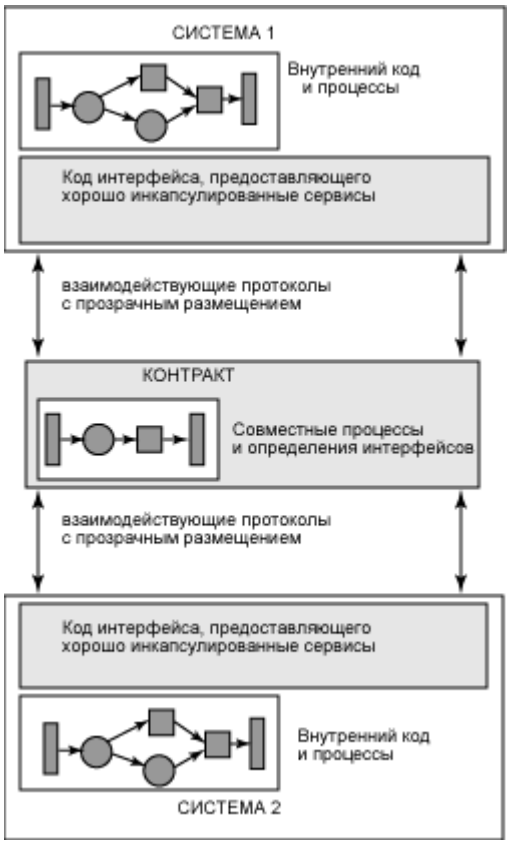
Роль ESB в структуре SOA

Хотя мы и не станем рассматривать в деталях определение SOA (см. раздел Ресурсы), все же здесь полезно будет собрать все принципы, с которыми соглашается большинство авторов определений SOA:

- Использование явно независимых от реализации интерфейсов для определения сервисов;
- Использование протоколов связи, усиливающих прозрачность расположения и функциональную совместимость;
- Определение сервисов, инкапсулирующих многократно используемые бизнес-функции.

Эти принципы иллюстрирует рисунок 1. Обратите внимание на то, что хотя технология Web-сервисов отлично согласуется с перечисленными принципами, это не единственная подобная технология.

Рисунок 1. Принципы SOA



Для реализации SOA принципы SOA должны поддерживаться и приложениями, и инфраструктурой. Чтобы создать приложение для SOA, необходимо напрямую или при помощи адаптеров создать интерфейсы сервисов для существующих или новых функций. Создание инфраструктуры на самом базовом уровне связано с предоставлением функций *маршрутизации* и доставки запросов сервисов нужному поставщику сервиса. Однако, кроме того, жизненно необходимо, чтобы инфраструктура поддерживала возможность *замены* одной реализации сервиса другой реализацией прозрачно для клиентов этого сервиса.

Для этого необходимо не только определить интерфейс сервиса в соответствии с принципами SOA, но и обеспечить возможность вызова сервиса клиентским кодом способом, независимым от размещения сервиса и используемых протоколов связи. Описанные функции маршрутизации и замены в числе многих других функций предоставляет шина ESB.

ESB поддерживает перечисленные функции взаимодействия сервисов и предоставляет интегрированную инфраструктуру связи, передачи сообщений и событий для их функционирования. Таким образом, шина объединяет основные используемые на сегодняшний день шаблоны корпоративной интеграции в одном объекте. ESB

предоставляет согласованную с требованиями корпорации инфраструктуру для SOA, для обеспечения требуемого уровня сервиса и управляемости, а также возможности функционирования в разнородной среде.

Далее в этой статье мы расскажем о роли ESB в структуре SOA, в том числе о функциях, предоставляемых шиной наряду с базовыми функциями маршрутизации и транспорта, описанными в модели зрелости для ESB (они тоже упомянуты в статье). Далее в этой статье мы расскажем о роли ESB в структуре SOA, в том числе о функциях, предоставляемых шиной наряду с базовыми функциями маршрутизации и транспорта, описанными в *модели зрелости для ESB* (они тоже упомянуты в статье).

Структура ESB

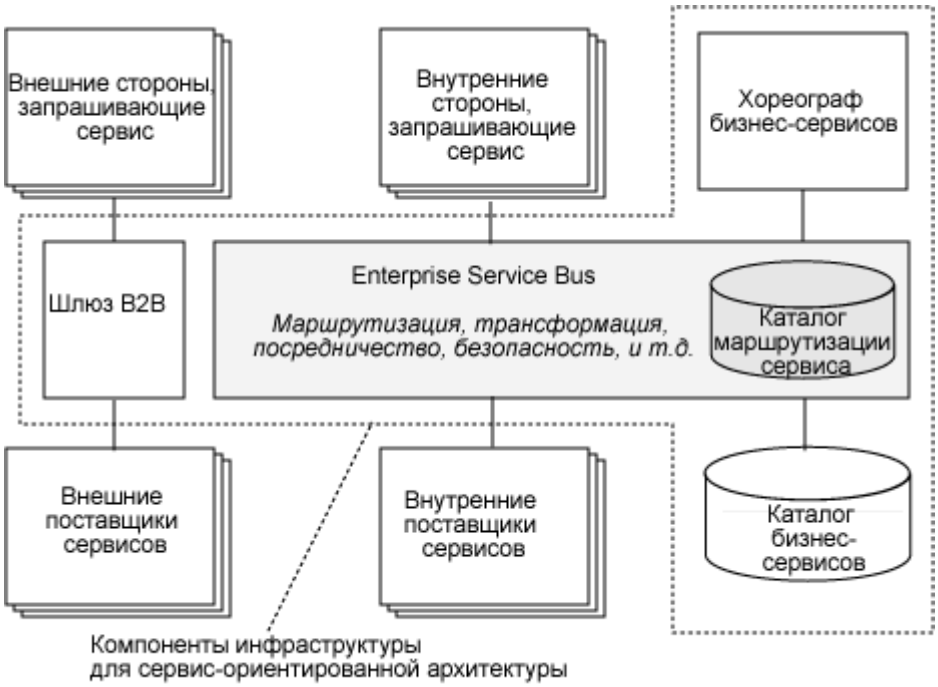
ESB иногда определяется как *распределенная* инфраструктура и противопоставляется другим технологиям, например, технологиям брокеринга сообщений, которые иногда называют технологиями типа "ступица и спицы" (*hub-and-spoke)*. Однако такое противопоставление неправомерно. При помощи упомянутых технологий решаются две разные задачи: *централизация управления* и *распределенность инфраструктуры*. Оба решения - и ESB, и технологии типа "hub-and-spoke" обеспечивают централизованное управление конфигурацией, в том числе маршрутизацию взаимодействий сервиса, назначение имен сервисам и т. п. Аналогично, оба решения могут быть развернуты в простой централизованной инфраструктуре или в более сложной, распределенной среде. Этот момент показан на рисунке 2.

Конечно, разные технологии будут иметь различные ограничения по физическому развертыванию поддерживаемых ими шаблонов - одни из них могут оказаться подходящими для очень масштабного распределения и поддержки интеграции в крупных географических областях, тогда как другие больше подходят для развертывания в локализованных кластерах и обеспечивают поддержку решений с высокой доступностью и масштабируемостью. Сопоставление требований к физической распределенности функций технологий, которые предполагается использовать, является важным аспектом разработки ESB. Кроме того, очень важно обеспечить возможность прирастающего расширения первоначального развернутой системы для отражения эволюционирующих требований, интеграции дополнительных систем или расширения географической доступности инфраструктуры.

Рисунок 2. Централизованное управление распределенной инфраструктурой ESB

Кроме того, следует обозначить позицию ESB по отношению к другим компонентам инфраструктуры SOA, в частности, компонентам службы каталогов Service Directory, Business Service Choreography и Business-to-Business (B2B) Gateway. Поскольку перечисленные выше принципы SOA не требуют обязательного наличия этих компонентов, давайте считать их необязательными компонентами. На рисунке 3 показана инфраструктура SOA, демонстрирующая отношения этих компонентов к ESB.

Рисунок 3: Роль ESB в структуре SOA



Для осуществления маршрутизации запросов сервиса ESB необходим особый *каталог маршрутизации сервиса*. Однако в SOA может также присутствовать отдельный *каталог бизнес-сервиса*, который, в самой простой форме, может представлять собой временный (используемый при разработке проекта) каталог, который используется для обеспечения многократного использования сервисов разработчиками организации. В представлении о Web-сервисах роль каталога бизнес-сервиса и каталога маршрутизации сервиса отводится каталогу UDDI, тем самым обеспечивается динамическое обнаружение и вызов сервисов. Такой каталог может рассматриваться как часть ESB, однако до тех пор, пока такие решения не получат достаточного распространения, лучше, чтобы каталог бизнес-сервиса существовал отдельно от ESB.

Функцией компонента Business Service Choreographer является компоновка *бизнес-процессов* из нескольких *бизнес-сервисов*; поэтому этот компонент вызывает сервисы через ESB, а затем предлагает бизнес-процессы клиентам как другие сервисы, также через ESB. Однако роль компонента Business Service Choreographer, технологии организации производственного процесса, в координации работы бизнес-процессов и сервисов идентифицирует этот компонент как невходящий в ESB, технологию инфраструктуры.

И наконец, функцией компонента B2B Gateway является обеспечение доступности сервисов каждой из двух или более организаций всем остальным организациям управляемым и безопасным способом. Полезно рассматривать такие компоненты как подключенные к ESB, но не входящие в ее состав. Хотя существуют шлюзовые *технологии,* предоставляющие функции, необходимые для реализации как компонентов B2B Gateway, так и ESB, само *назначение* компонента B2B Gateway отделяет его от ESB. Действительно, для выполнения своих функций этому компоненту могут потребоваться дополнительные средства, например, инструменты управления партнерскими

взаимоотношениями, которые не являются частью ESB и не обязательно поддерживаются технологиями ESB.

Модель производительности ESB

Таблица 1 обобщает и классифицирует некоторые функции ESB, описанные в имеющейся литературе (см. раздел Ресурсы). Одни из этих функций являются простыми, другие, например, функции автономности и интеллектуальные функции, представляют собой значительный шаг к операционной среде по требованию (On Demand). Важно понимать, что для большинства существующих сценариев использования ESB необходимыми являются только некоторых из этих функций из некоторых категорий. О *минимальном* наборе функций, необходимых для реализации ESB, мы поговорим в разделе *Минимальный набор функций ESB для SOA*.

Таблица 1. Функции ESB, описанные в специальной литературе

Связь	**Взаимодействие сервисов**
• Маршрутизация; • Адресация; • Технологии, протоколы и стандарты связи (например, IBM® WebSphere® MQ, HTTP и HTTPS) • Публикация/подписка; • Ответ/запрос; • Запустил-и-забыл, события; • Синхронный и асинхронный обмен сообщениями.	• Определение интерфейса сервиса (например, язык WSDL (Web Services Description Language, язык описания Web-сервисов); • Поддержка возможности замены реализации сервиса; • Модель организации сервиса обмена сообщениями, необходимая для связи и интеграции (например, SOAP или модель связующего ПО корпоративной интеграции приложений (EAI, enterprise application integration); • Каталог сервиса и обнаружение сервиса.
Интеграция	**Качество сервиса**
• База данных; • Агрегация сервиса; • Адаптеры для имеющихся систем и приложений; • Обеспечение подключения к связующему ПО EAI; • Отображение сервиса; • Преобразование протоколов; • Среда сервера приложений (например, J2EE и .NET); • Интерфейс языка прикладного программирования для вызова сервиса (например, Java и C/C++/C#).	• Транзакции (Неделимые транзакции, Компенсация, WS-транзакция); • Различные парадигмы гарантированной доставки (например, WS-ReliableMessaging или поддержка связующего ПО EAI).
Безопасность	**Уровень сервиса**
• Аутентификация;	• Производительность;

• Авторизация; • Невозможность отказа от авторства; • Конфиденциальность; • Стандарты обеспечения безопасности (например, Kerberos и WS-Security).	• Пропускная способность; • Доступность; • Прочие непрерывные меры, которые могут стать основой контрактов или соглашений.
Обработка сообщений	**Управление и автономность**
• Закодированная логика; • Логика, основанная на контенте; • Преобразование сообщений и данных; • Проверка корректности; • Посреднические; • Тождественное отображение объектов; • Обогащение данных.	• Инициализация и регистрация сервиса; • Ведение журналов, измерения, мониторинг; • Обнаружение; • Интеграция с инструментами управления и администрирования системы; • Самонаблюдение и самоуправление.
Моделирование	**Интеллектуальные функции инфраструктуры**
• Моделирование объектов; • Общие модели бизнес-объектов; • Библиотеки форматов данных; • Открытые или закрытые модели для интеграции B2B; • Инструменты разработки и развертывания.	• Бизнес-правила; • Управляемое политиками поведение, особенно для функций уровня сервиса, обеспечения безопасности и качества сервиса (например, WS-Policy); • Распознавание шаблона.

Многие из этих функций могут быть реализованы при помощи соответствующих технологий или с использованием открытых стандартов. Но технологии, претендующие на использование в реализации ESB, могут в значительной степени различаться по характеристикам производительности, масштабируемости и доступности, а также по тому, какие функции ESB и открытые стандарты они поддерживают. По этим причинам и вследствие того, что некоторые из значимых стандартов были разработаны недавно или все еще находятся в стадии разработки, многие критически важные решения по реализации ESB в настоящее время сопряжены с поиском компромиссов между поддержкой зрелых, сложившихся технологий и менее зрелых открытых стандартов.

В данной серии статей я не буду подробно рассказывать о каждой из этих категорий функций. Скорее мы сосредоточимся на тех из них, которые имеют значение для принятия решений по поводу выбора подхода к внедрению или реализации ESB. В частности, в следующем разделе мы остановимся на том, какой минимальный набор функций необходим для поддержки SOA реализацией ESB.

Минимальный набор функций ESB для SOA

Если для большинства SOA-сценариев имеет значение только подмножество перечисленных ранее функций, мы можем поставить следующий вопрос: какие функции входят в *минимальный* набор функций, необходимых для того, чтобы реализовать ESB? Чтобы ответить на этот вопрос, мы рассмотрим самые распространенные элементы определения ESB, по поводу которых не существует особых разногласий:

- ESB представляет собой логический компонент архитектуры, который предоставляет инфраструктуру интеграции, согласующуюся с принципами SOA;
- Согласно принципам SOA, необходимо использование независимых от реализации интерфейсов, протоколов связи, усиливающих прозрачность размещения и функциональную совместимость, а также определения сервисов, которые являются относительно малодетализированными и инкапсулируют функцию, допускающую многократное использование;
- ESB может быть реализована как распределенная неоднородная инфраструктура;
- ESB предоставляет средства для управления инфраструктурой сервисов и дает возможность работать в современной распределенной разнородной среде.

В таблице 2 приводится минимальный набор функций ESB, выбранных с учетом этих принципов.

Таблица 2. Минимальный набор функций ESB

Связь	**Интеграция**
• Маршрутизация и адресация сервисов для обеспечения прозрачности размещения; • Функция администрирования для управления адресацией и именованием сервисов; • По меньшей мере одна из форм парадигмы обмена сообщениями (например, запрос/ответ, публикация/подписка и т. п.); • Поддержка по меньшей мере одного транспортного протокола, который является или может стать общедоступным.	• Поддержка нескольких средств интеграции с поставщиками сервиса, например, коннекторов Java 2, Web-сервисов, асинхронного обмена сообщениями, адаптеров и т. п.
Взаимодействие сервисов	
Открытая и независимая от реализации модель обмена сообщениями и организации интерфейсов, которая должна изолировать код приложения от специфических условий маршрутизации сервисов и транспортных протоколов, а также обеспечение возможности замены реализации сервиса	

Обратите внимание на то, что минимальный набор функций *не требует использования каких-либо определенных технологий*, например, связующего ПО EAI, Web-сервисов, J2EE или XML. Вполне вероятно, что такие технологии будут использоваться, поскольку они отвечают требованиям, но это не является обязательным. И наоборот, минимальный набор функций почти, если не полностью, обеспечивается простым использованием SOAP/HTTP и WSDL.

- Адресация URL и существующая инфраструктура HTTP и DNS предоставляют инфраструктуру "шины", обеспечивающей маршрутизацию сервисов и прозрачность размещения;
- SOAP/HTTP поддерживает парадигму обмена сообщений *запрос-ответ*;

- Транспортный протокол HTTP является широко доступным;
- SOAP и WSDL представляют собой открытую, независимую от реализации модель организации интерфейса и обмена сообщениями сервисов.

Тем не менее, использование SOAP/HTTP и WSDL в простейшей форме в действительности обеспечивает только интеграцию от точки до точки и не предоставляет следующих ключевых функций, необходимых для ESB:

- Отсутствует *функция администрирования* для управления адресацией и наименованием сервисов Имена сервисов контролируются в индивидуальном порядке каждым адаптером, поэтому управление маршрутизацией сервиса распределяется между адресами, вызываемыми клиентами сервиса, инфраструктурой HTTP и именами сервисов, присваиваемых адаптерам;
- Хотя этот подход зависит от деталей реализации, он не способствует обеспечению замены реализации сервиса; код запросчика сервиса (возможно, сгенерированный инструментами разработки), часто будет напрямую привязан к специфической реализации поставщика сервисов через специфические протоколы по специфическим адресам. Замена реализации сервиса другой реализацией потребует изменений в коде приложения и его повторного развертывания.

Безусловно, для многих или даже для большинства сценариев необходимы дополнительные функции, которые со временем станут все более распространенными. В частности, следующие типы требований, вероятно, приведут к использованию более сложных технологий, сейчас или в будущем:

- Функции обеспечения качества сервиса и уровня сервиса;
- Концепции SOA более высокого уровня - service choreography, каталог, преобразование и т. д.;
- Требования операционной среды по требованию (On Demand), такие как функции автономности и управления, а также интеллектуальные функции инфраструктуры;
- По-настоящему неоднородные операции в нескольких сетях, с несколькими протоколами и несколькими доменами с разными моделями владения.

Проблемы безопасности, связанные с ESB

В этой статье не предполагалось рассматривать сами требования к безопасности, однако, они могут иметь значение при выборе технологии ESB. Например, если нет необходимости в аутентификации и авторизации запросов к серверу, то выбор технологий может быть очень обширным. Если требуется обеспечить определенный уровень безопасности, что более вероятно, то важно оценить, какой стиль обеспечения безопасности будет приемлемым. Например:

1. Будет ли приемлемым уровень безопасности инфраструктуры связи при использовании взаимной аутентификации по протоколу безопасных соединений Secure Socket Layer EAI между серверами связующего ПО EAI или при использовании протокола HTTPS?
2. Будет ли достаточно индивидуальной, от точки к точке, системы безопасности между системами-участниками, или необходима сквозная модель обеспечения безопасности? Например, есть ли необходимость в распространении идентификационной информации клиента через промежуточные системы, например, брокеры, конечным поставщикам или реализациям сервиса?
3. Будет ли приемлемой безопасность на прикладном уровне, например, может ли клиентский код выполнить базовую аутентификацию HTTP по идентификатору

пользователя и паролю, или сможет ли он передать эту информацию сервису как данные приложения?

4. Требуется ли соответствие механизма безопасности стандартам обеспечения безопасности, например, Kerberos или WS-Security?

Хотя возможны все перечисленные подходы к обеспечению безопасности, рекомендуется использование соответствующих стандартам (например, WS-Security) функций обеспечения безопасности при поддержке инфраструктуры и связующего ПО.

Однако эти стандарты появились сравнительно недавно, и поддержка их программными продуктами пока находится в фазе развития, особенно в случаях, связанных с обеспечением функциональной совместимости. Таким образом, одним из приоритетов архитектуры ESB должно стать утверждение требований к обеспечению безопасности на ранних фазах разработки, чтобы их можно было учесть при выборе технологии реализации.

Заключение

В данной статье рассказывалось о самых общих принципах SOA и их связи с технологией Web-сервисов. Исходя из этих принципов автор утверждает, что компонент инфраструктуры, предоставляющий функцию маршрутизации, должен обеспечить взаимодействие сервисов между собой, а также поддержку замены одной реализации сервиса другой реализацией. Эти функции, в числе многих других, реализуются посредством ESB.

ESB предоставляет распределенную инфраструктуру и функции централизованного управления, для чего необходим каталог маршрутизации сервиса и кроме того, возможно, каталог бизнес-сервиса. Компонент Business Service Choreographer вызывает сервисы из ESB, а затем представляет процессы как новые сервисы через ESB.

Среди множества функций, предоставляемых ESB, имеются функции:

- Связи;
- Взаимодействий сервисов;
- Интеграции;
- Обеспечения качества сервиса;
- Безопасности;
- Обеспечения уровня сервиса;
- Обработки сообщений;
- Управления и автономии сервиса;
- Моделирования;
- Интеллектуальные функции инфраструктуры.

Из этих разных функций автор сформировал минимальный набор для создания ESB, обеспечивающей связь, интеграцию и взаимодействие сервисов.

В следующей статье серии мы рассмотрим распространенные сценарии, подходящие шаблоны решений для сценариев и поговорим о самых распространенных проблемах, связанных с этими сценариями.

Благодарности

Эта статья вряд ли вышла в свет, если бы автор не обсуждал свои идеи со следующими людьми: Бет Хатчисон (Beth Hutchison), Рейчел Рейниц (Rachel Reinitz), Олаф Циммерман (Olaf Zimmerman), Хелен Уайли (Helen Wylie), Кайл Браун (Kyle

Brown), Марк Колан (Mark Colan), Джонатан Адамс (Jonathan Adams), Пол Фремантл (Paul Fremantle), Кейт Джонс (Keith Jones), Пол Вершурен (Paul Verschueren), Дэниэл Стэрмен (Daniel Sturman), Скотт Косби (Scott Cosby), Дейв Кларк (Dave Clarke), Бен Манн (Ben Mann), Луиза Гиллис (Louisa Gillies), Эрик Хернесс (Eric Herness), Билл Хасселл (Bill Hassell), Гуру Васудева (Guru Vasudeva), Карим Юсуф (Kareem Yusuf), Кен Уилсон (Ken Wilson), Марк Эндреи (Mark Endrei), Норберт Биберштейн (Norbert Bieberstein), Крис Нотт (Chris Nott), Алан Хопкинс (Alan Hopkins) и Ярослав Данчич (Yaroslav Dunchych).