INFM309

Oracle & Java

01. Системни архитектури

Александър Станев
alex at stanev.org
http://stanev.org

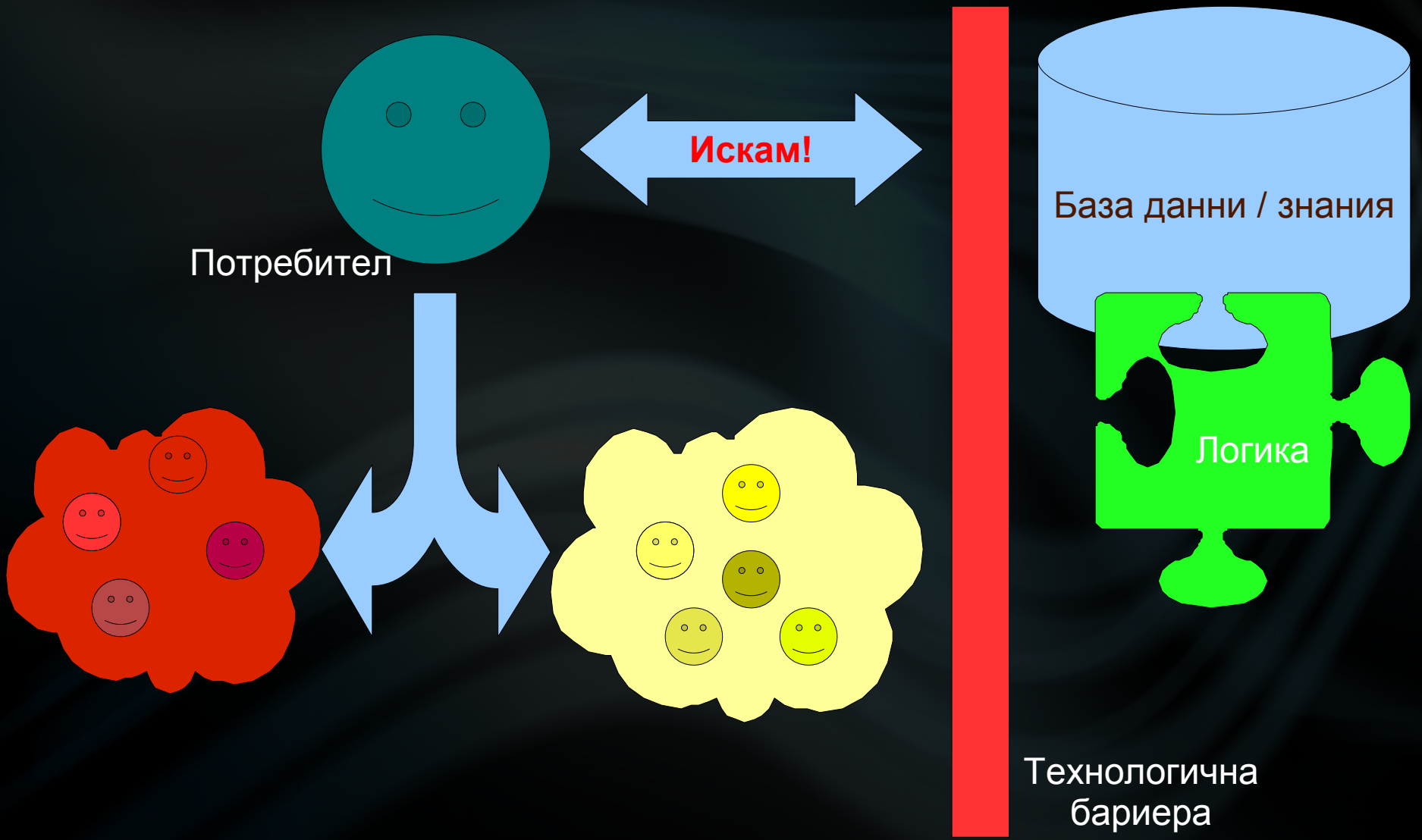
Понятия - I

- Функция на информационните системи
- ***Информационните системи*** са изградени от различни компоненти (*модули*), свързани помежду си чрез комуникационни *интерфейси*
- ***Актьор*** – потребител на системата, извършващ действия в нея

Понятия - II

- ***Модул – компонент, изпълняващ определена логическа задача***
  - Стандартни
  - Специализирани (външни)
- ***Интерфейс* – връзката между модулите, чрез която си обменят информация – формат на данните**
- ***Протокол*** – начина на предаване на информация между модулите

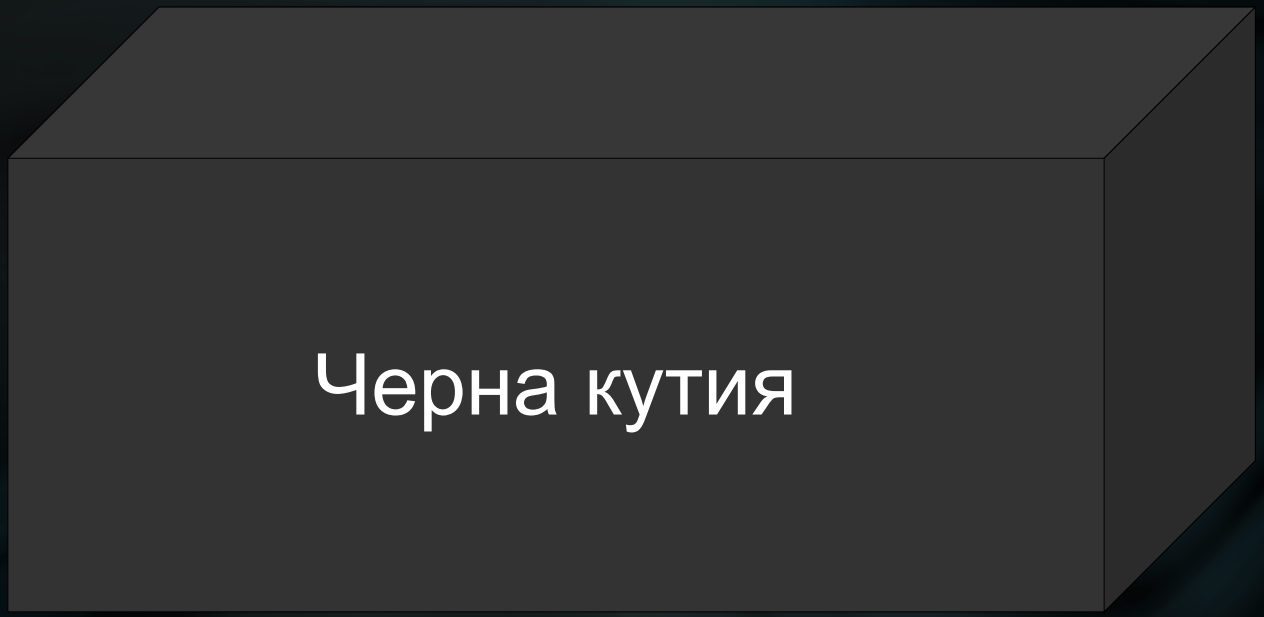
Логически модел на информационна система



ИС от близкото минало

- Защо се занимаваме с това?
- Хомогенни системи, енкапсулирали в себе си данните и логиката за тяхното управление
- Технологичните решения изискват добро познаване и правилен избор на структурите данни, алгоритмичните решения и базовата хардуерна архитектура
- Примери

ИС от близкото минало - схема



ИС от близкото минало - плюсове

- Висока производителност
- Ниски системни изисквания – увеличава ROI на хардуера за сметка на понижаване за софтуера
- Обикновено потребителите са свикнали с интерфейса (колкото и да ни се струва странен, неудобен и крив) и се опитват да го трансферират в новите системи

ИС от близкото минало - минуси

- Черна кутия
- Подчиняват се бизнес-процесите на технологичните нужди
- Липсва ясно разделение между отделните модули
- В общия случай програмистите са разработили стандартните модули (custom solution)
- Работа в мрежова среда?
- Трудна миграция (в някои случаи плюс ;)

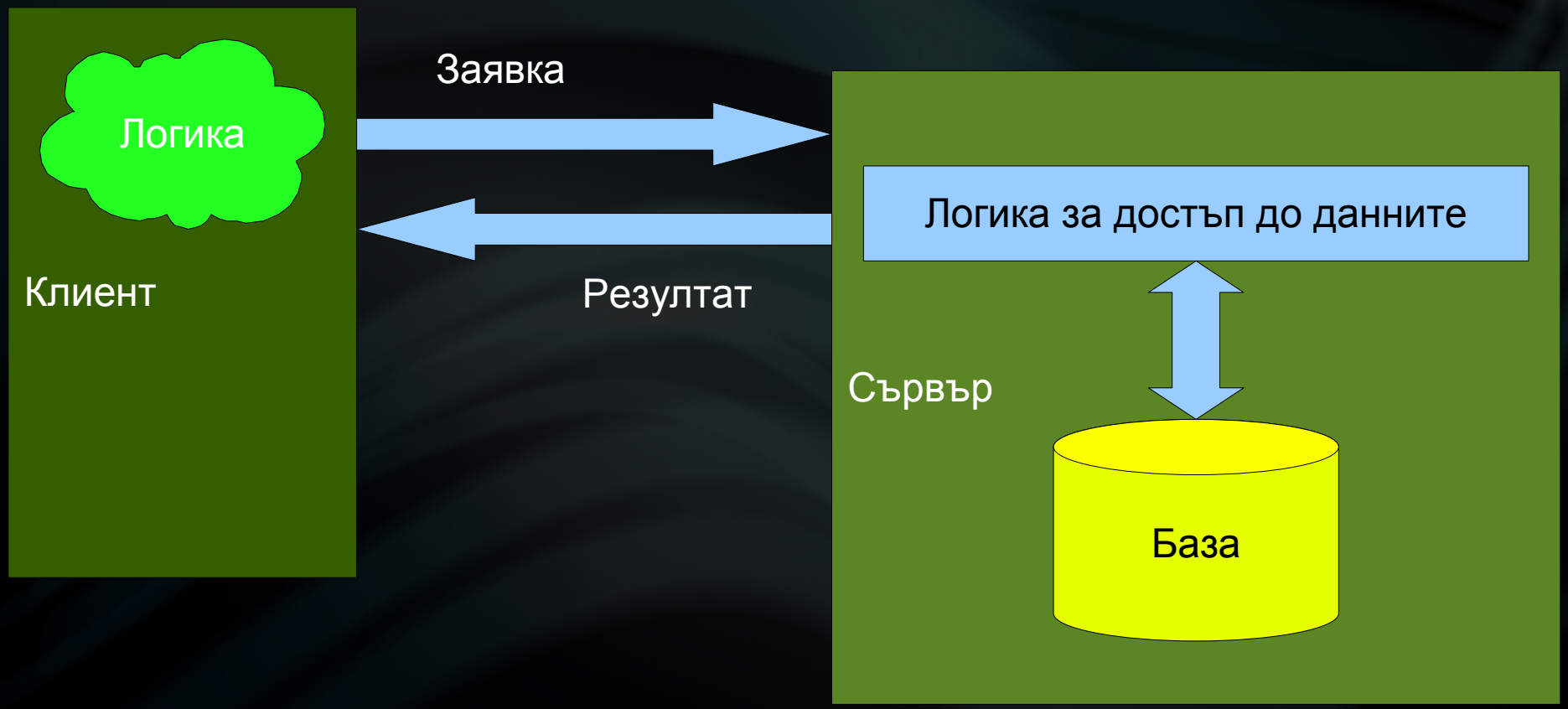
Клиент - Сървър

- Модулът при клиента комуникира с потребителя на системата
- Сървъра получава заявки от клиента, обработва ги и връща резултат
- Резултата се представя в подходяща за потребителя форма
- Логиката на системата обикновено е разпределена, но може и да е концентрирана в един от модулите

Клиент – Сървър описание

- Модулът при клиента комуникира с потребителя на системата
- Сървъра получава заявки от клиента, обработва ги и връща резултат
- Резултата се представя в подходяща за потребителя форма
- Логиката на системата обикновено е разпределена, но може и да е концентрирана в един от модулите

Клиент – Сървър примерна схема



Клиент – Сървър плюсове

- Интеграция с работната среда
- Скъсяване на критичният път
- Подходящо за по-малки проекти*
- Подходящо за по-малки организации*

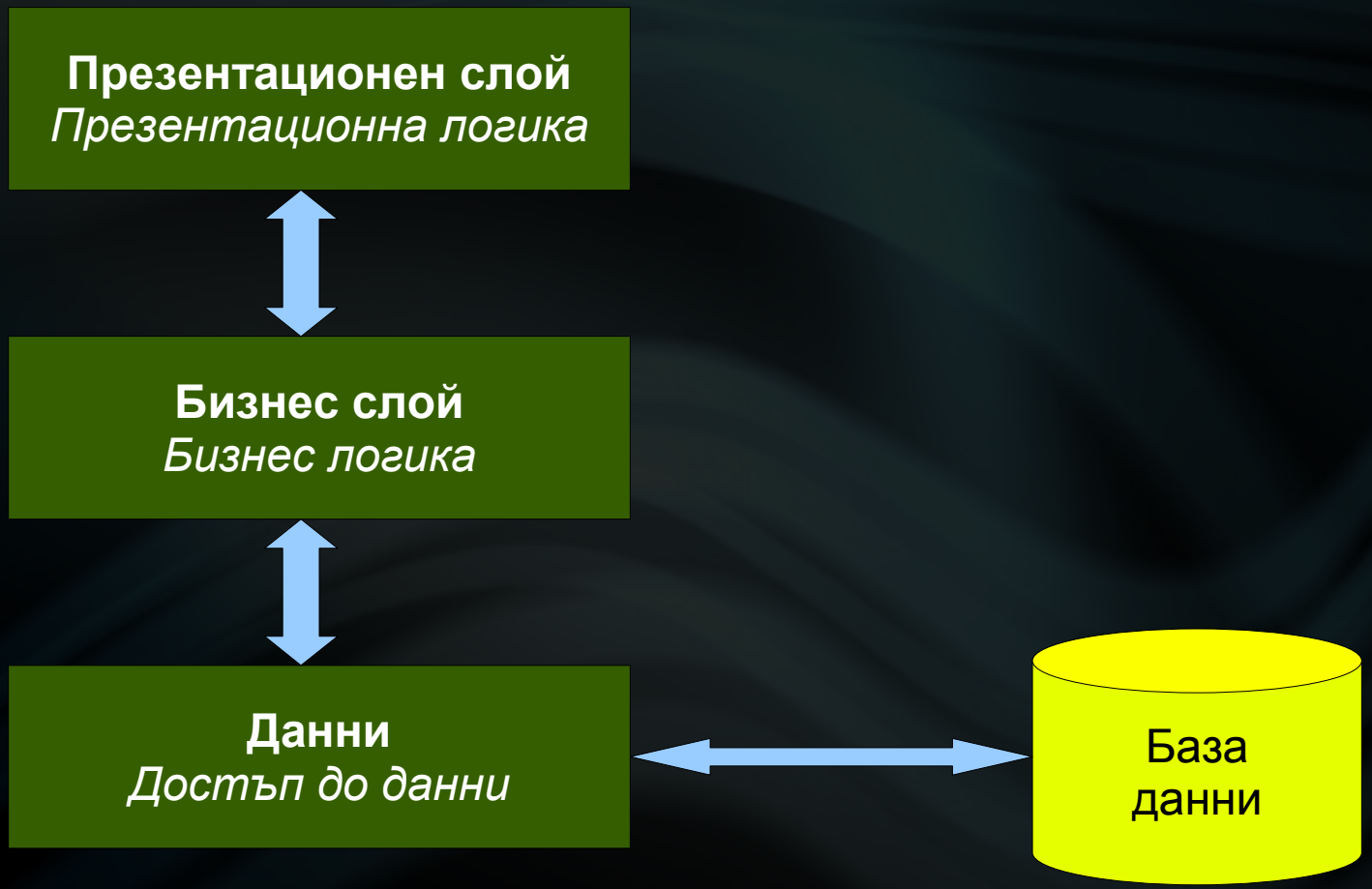
Клиент – Сървър минуси

- Трудно балансиране на натоварването
- Необходимост за управление версиите на клиента
- Многоплатформеност
- Сигурност и проследимост на процесите

Трислойна архитектура: Еволюция

- Стандартна 3tier архитектура:
  - (Лек) клиент – интеракция с потребителя
  - Приложен сървър – енкапсулира системната бизнес-логика
  - База данни – съхранение на информация
- Model-View-Controller (MVC) – често се бърка, но не е!

Трислойна архитектура схема



Трислойна архитектура плюсове

- Разделението на слоевете опростява тяхното управление (Divide et impera!)
- Добавянето на нова функционалност е по-лесно, особено ако е само в един от слоевете
- Архитектурата е силно разпространена и се намират специалисти (!)
- It's not a rocket science.

Трислойна архитектура минуси

- Според технологията и средствата за разработка, може лесно да се „залитне“ към смесване на слоевете
- Изисква по-сериозно планиране и не винаги дава възможност да се използва пълния капацитет на използваните компоненти
  Пример: „базата данни само за storage“, но в Oracle можем да направим много повече

Enterprise Application Integration (EAI)

- Използва се за интеграция на приложения в една обща система
- Слоевете зависят от избрания механизъм на управление:
  - Leading node
  - Peer-to-peer
- Ингтеграционни решения
  - Mediation (интегрционна шина / EIB)
  - Federation (портално решение / PS)
  - Смесен тип

EAI – Нива на интеграция

- Данни
- Бизнес процеси
- Независим вход/изход (vendor independence)
- Портал (common facade)

EAI – Топологии

- Hub-and-spoke
  Всички заявки (messages) минават през централен модул, който ги разпределя
- Bus
  Организира се обща шина за съобщения, по която различните приложения си комуникират чрез стандартни заявки (enterprise message model)

EAI – Акценти - I

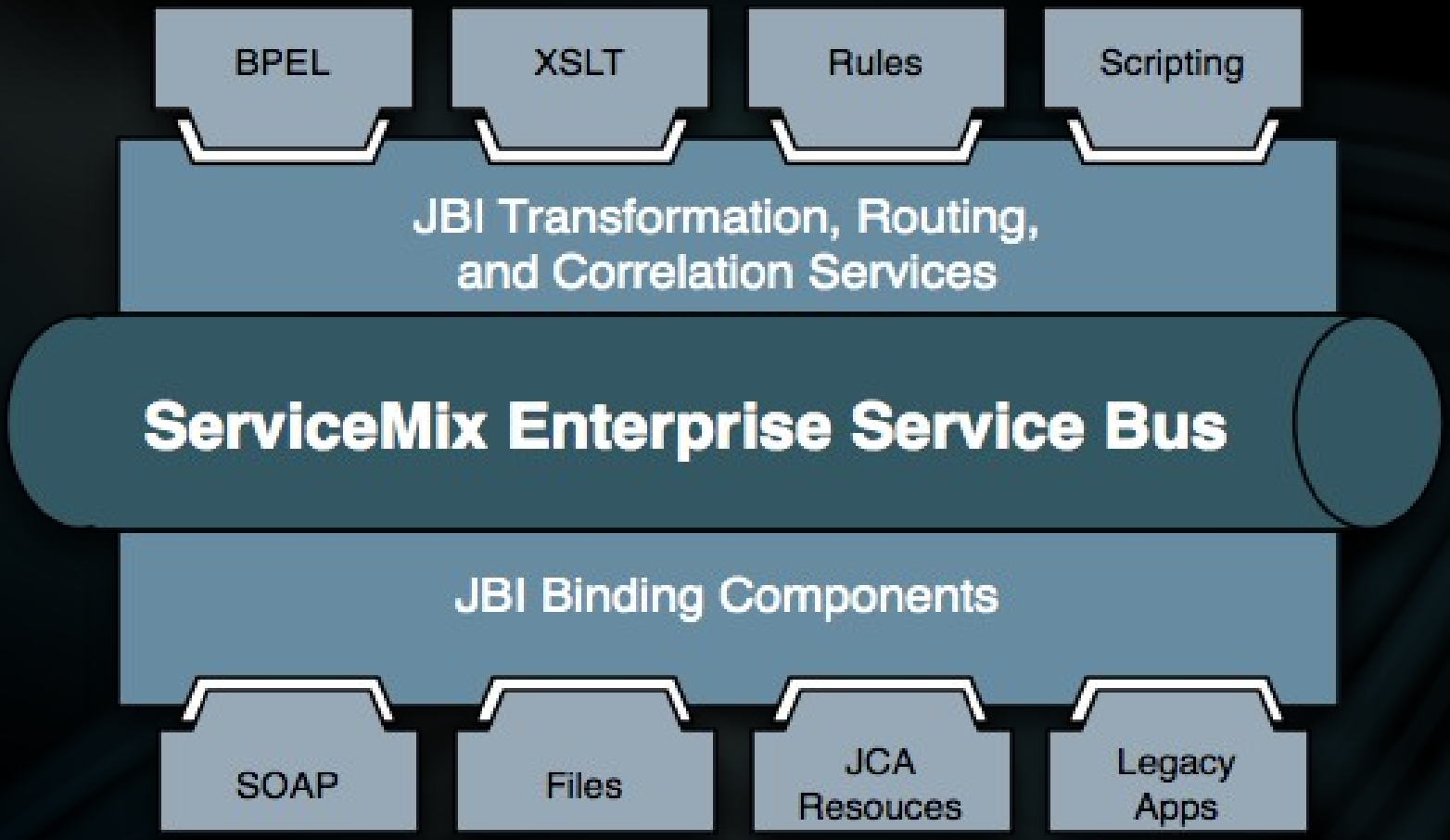
- Не зависи от използваната технология – интегрира Java, .NET, legacy приложения
- Това дава възможност да се удължи живота на модули (на практика системи), недостигнали необходимото ниво на ROI
- Ако промените в общата система са инкрементални е възможно постигането на zero downtime – всички останли освен засегнатите модули продължават да работят

EAI – Акценти - II

- Много по-трудно управление
- Необходимост за имплементация на общия комуникационен интерфейс
- Необходимост от съобразяване на множество състояния, за които отделните приложения не са били подготвени
- Повишена латентност на отделните модули поради изчаквания по шината
- Обикновено се нуждаем от още хардуер

EAI – пример за bus integration

Service Oriented Architecture (SOA) + Event Driven Architecture (EDA)



ServiceMix, проект на Apache [http://servicemix.apache.org]
базиран на Java Bussiness Integration, Java Messaging Queue, Java Cryptography Architecture, и т.н.

Service Oriented Architecture (SOA)

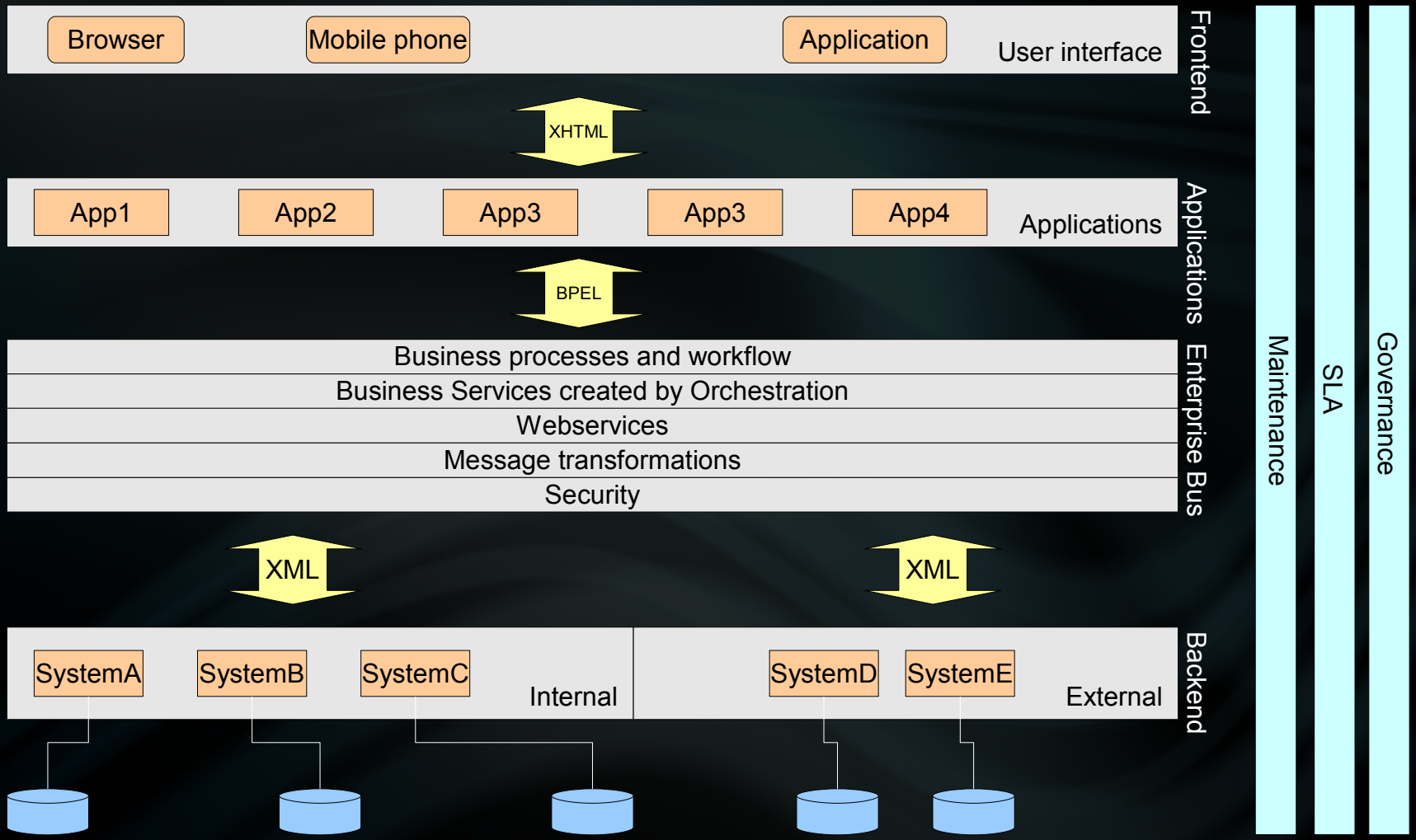
- Вариант на EAI, дефиниращ модела на комуникация между модулите (nodes), както и редица други параметри на архитектурата
- Бизнес процесите се пакетират като услуги с дефиниран вход/изход

- WSDL – за описание на услугата
- Използва SOAP messaging за транспорта
- UDDI регистър на услугите

Допълнителни технологии

- За пълното използване на възможностите на съвременните архитектури са разработени нови технологии, опростяващи (дават възможност на бизнес хората) връзката между реалния свят и виртуалния
  - BPEL (language and processor)
  - Service orchestration
  - Message queues
  - Event ques
  - ...

Service Oriented Architecture (SOA) - схема



* Използваната схема е от http://ttlnews.blogspot.com/

Защо правим всичко това!?

- За да доставим на бизнеса необходимата му информация
- За да доставим на бизнеса необходимата гъвкавост
- За да дадем възможност за пълна проследимост и измерване на процесите
- За да постигнем висок ROI в организацията
- **За да правим пари***

* консултанти vs. потърпевши

Проблеми, тенденции, наблюдения

- ***Принцип на Усложняване***
  Всичко става все по-сложно
- ***Необходимост за по-добра Производителност***
  Ако не го направим по-добре, защо го правим въобще?
- ***Сигурност***
  Конкуренцията не спи; случайните „минувачи“ - също
- ***Обем***
  Данните нарастват геометрично

Страница на курса

Слайдовете, материалите и допълнителна информация за курса могат да бъдат открити на адрес:

http://alex.stanev.org/training/nbu-infm309

Въпроси

Слайдовете, материалите и допълнителна информация за курса могат да бъдат открити на адрес:

http://alex.stanev.org/training/nbu-infm309

TPILB.

This page intentionally left blank.