

**Going SOA with CA Plex and Websybian
TransacXML**

Speakers

- Søren Madsen
Chief Consultant, Soft Design A/S

 - Anne-Marie Arnvig
Communications Manager, Websyidian A/S
-

Agenda

- SOA vs. Web Services
 - What is a service?
 - What is Service Oriented Architecture?
 - The SOA Benefits
 - SOA in the Real World
 - Services and interfaces
 - Going SOA with Plex and Websyodian: demo of Websyodian TransacXML
 - Questions
-

SOA vs. Web Services

SOA ≠ Web Services

You can have SOA without Web Services

You can have Web Services without SOA

What is a service?

What is a service?

Definition taken from the field of Service Design:

A service consists of one or a series of contact points where some kind of interaction or interchange occurs:

- Ordering a sandwich at the counter at the deli
 - Calling your local tax authorities for instructions
 - Retrieving data on a web site
 - ...
-

What is a service?

A service is a way of organizing or structuring a series of actions/interchanges in order to lead the service receiver to his/her goal.

What is Service Oriented Architecture?

Service oriented architecture

- SOA is more a mindset than an actual technology.
 - The general idea is to work smarter not harder by constructing and structuring systems as sets of building blocks that can all be fitted with each other to create new facilities.
 - The building blocks are services.
 - The notion is that services built correctly are so loosely coupled that they can be shared via the Internet and used for other applications than the one they were originally designed for.
-

The SOA Benefits

Object reuse

- SOA breaks down an application into small independent pieces of functionality.
 - Properly made, these services can be reused in multiple applications.
-

Parallel development

- Since services are independent of each other, and “contracts” between services are pre-defined, the services can more easily be developed in parallel – this shortens the software development life cycle considerably



Focused developer roles

- A service is a separate implementation independent of other services.
 - Developers can focus completely on implementing and maintaining the individual services.
-

Platform independence

Built correctly the services can be published and consumed across development and operating platforms:

- Leveraging existing applications
 - Building additional functionality without having to rebuild the entire application
 - Integrating applications with those of partners.
-

Greater testability

Small, independent items are easier to test and debug than monolithic applications - this means more reliable software - faster!

Easier to scale and higher availability

The services can be moved to a more powerful server to service more consumers if required.

Also, there can be multiple instances of the service running on different servers.

Service Oriented Architecture

The supposed benefits:

- Code reuse
 - Parallel development
 - Focused developer roles
 - Platform independence
 - Greater testability
 - Better scalability
 - Higher availability
-

SOA in the Real World

SOA in the Real World

It all sounds really fine and easy – in theory that is.

Because we all have applications that are not structured like building blocks, and real life doesn't always permit us to work that disciplined.

SOA in the Real World

Can we use the SOA mind set at all in the real world?



Luckily you work with CA Plex

Absolutely!

CA Plex developers have a head start compared to traditional coders. With model based development you can profit from these benefits.

Plex ensures:

- modular separation of business processes
 - that processes and directories are kept tidy and can be reused by our colleagues effortlessly
-

SOA in the Real World

You don't have SOA yet...

... but you are much closer than you might think.

Going SOA with CA Plex and Websyidian

- Consider how you want to structure actions/interchanges as services.
 - Use the general idea of building blocks with Web Services.
 - Start thinking in terms of independency regarding platforms, applications etc. using the Web Services standards: XML, WSDL etc.

 - Extend your possibilities and applications
 - Get immediate benefit from the SOA idea
-

A practical approach

Søren Madsen, Soft Design A/S

- Chief Consultant

- Worked with CA 2E since 1990
- CA Plex since 1996

- Speaker at:

- CA WORLD and Plex/2E user conferences since 2004
 - COMMON US since 2007
-

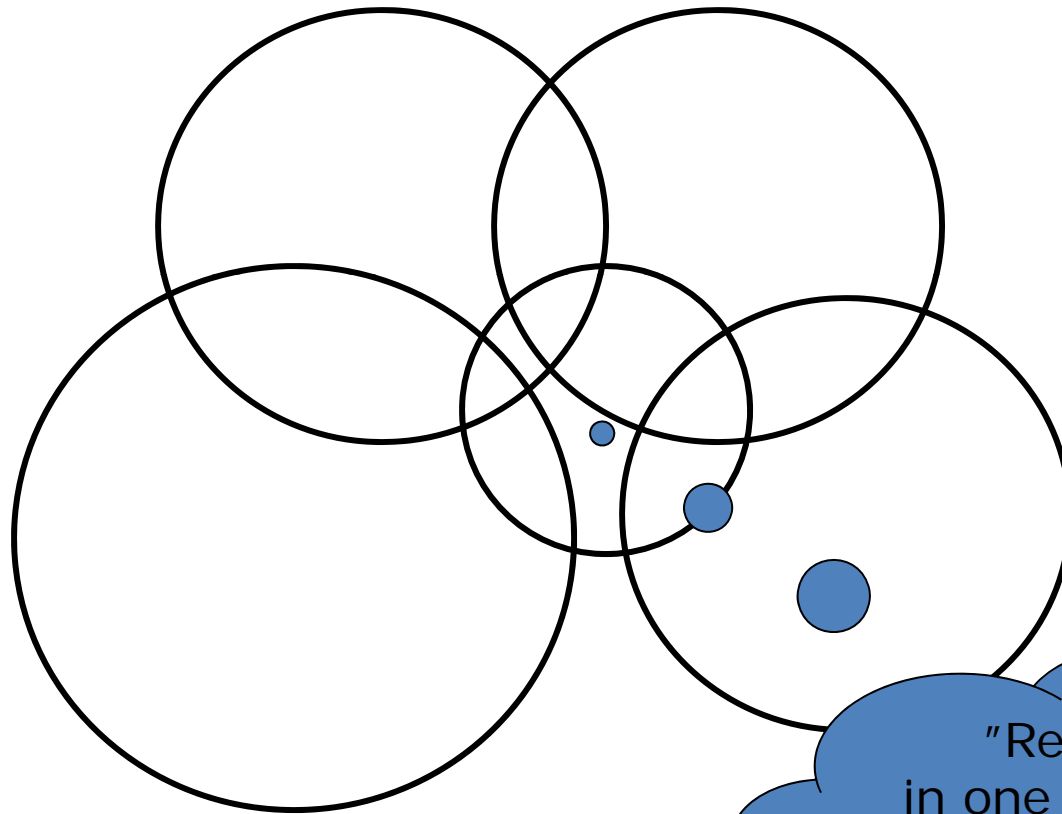
Services

Why all this talk about services

- You probably already made "services" yourselves
 - Moduls, API's, ServiceFunctions, or whatever you call them.



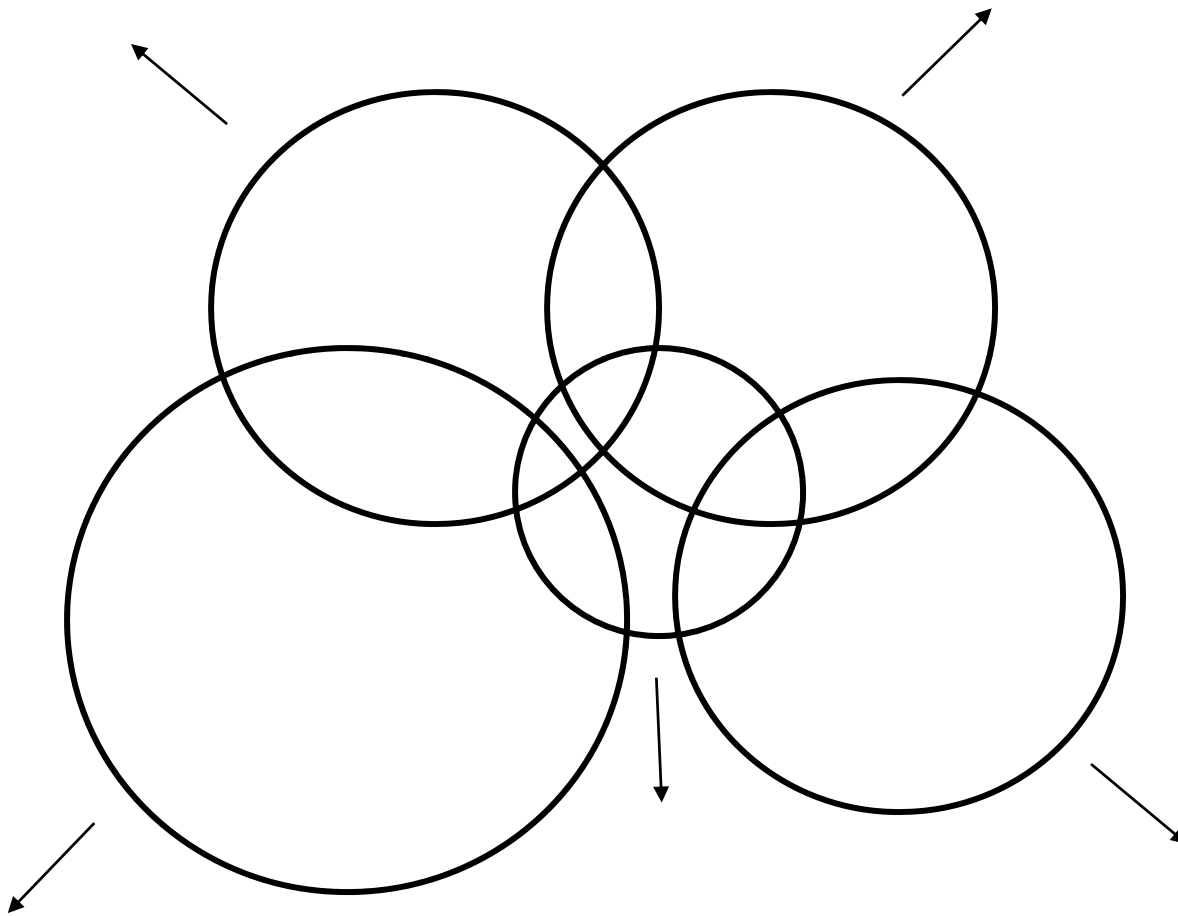
Traditional development and CA Plex systems



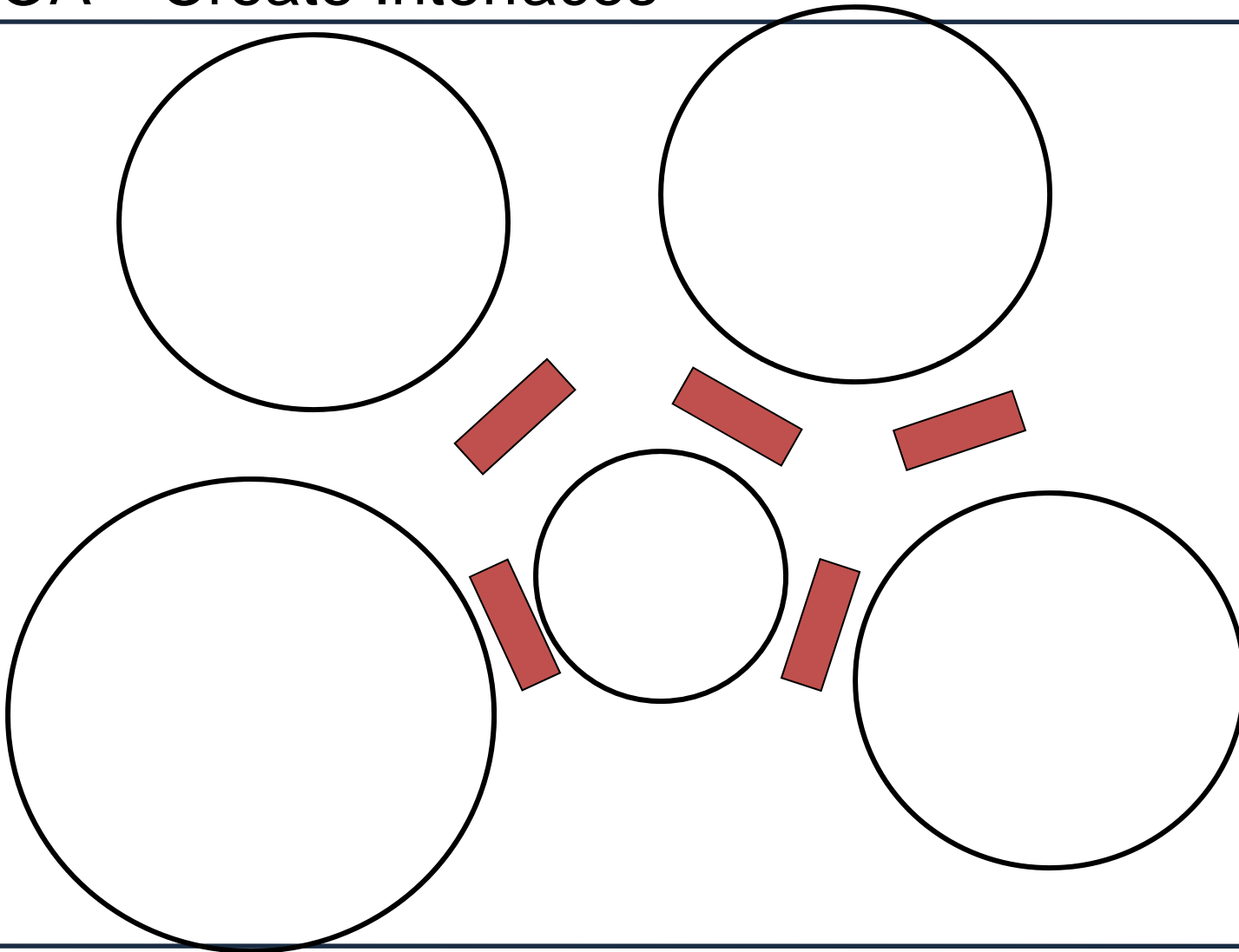
Every circle illustrates a system

"Read an item" in one of our solutions is called by 109 functions => High maintenance costs

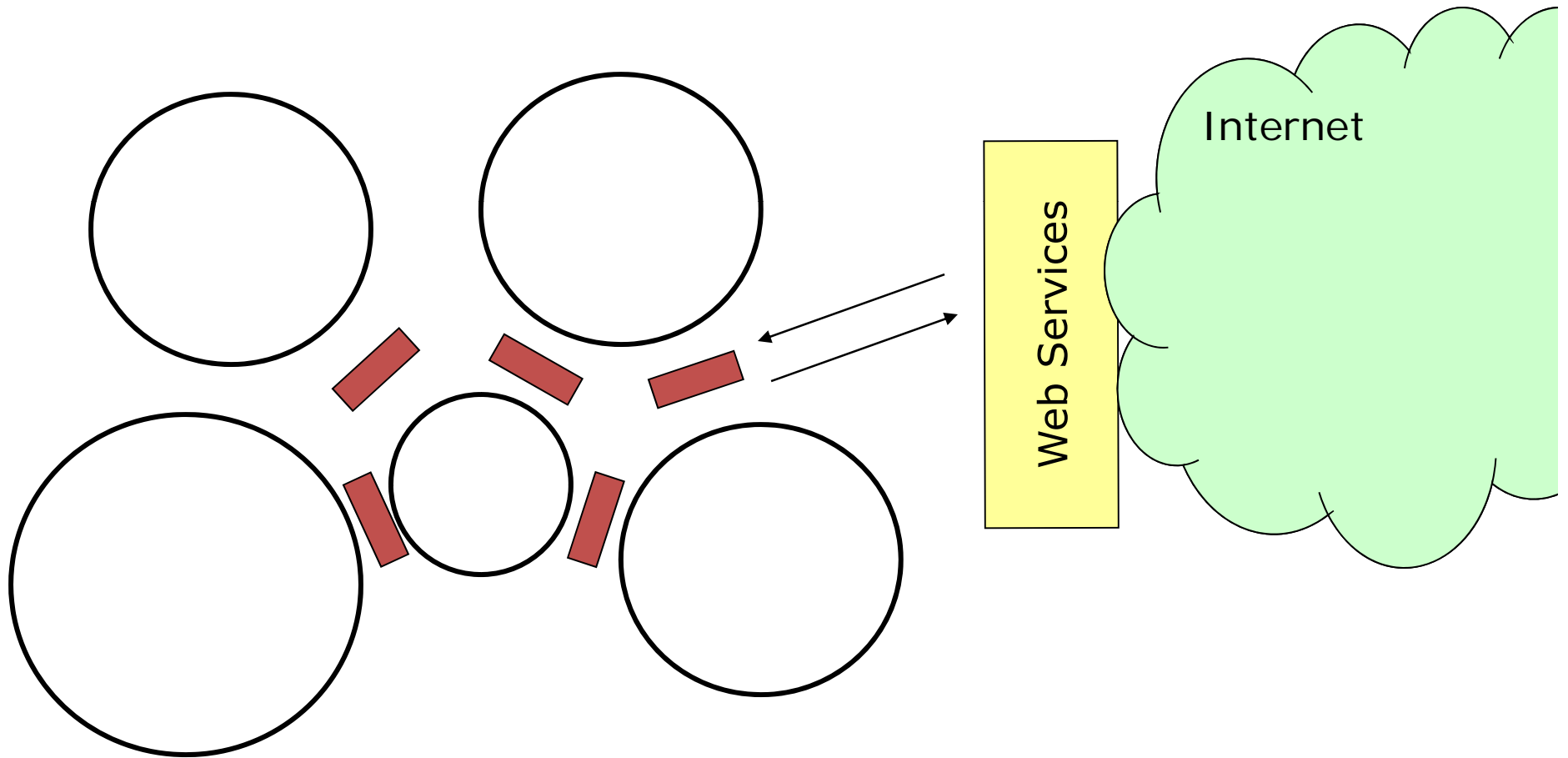
SOA – Separate Our Application Areas



SOA – Create Interfaces



Web Services - when relevant



Interfaces

- Has to be stable
 - I.e. a change must NEVER influence a caller.
 - Underlying logic to return an answer can be adjusted as long as the answer remains the same format.
 - If an interface is to be adjusted, a new interface is created
 - So the existing interface is stable.
 - Caller has to change to the new interface if he needs to.



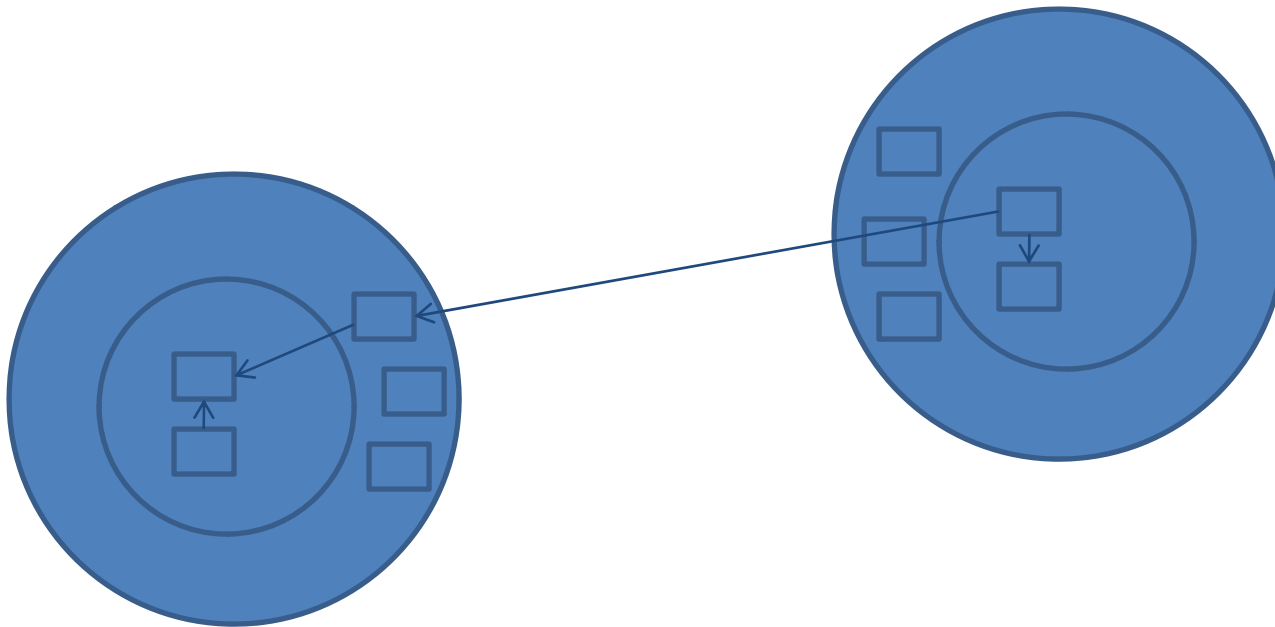
Interfaces

- Has to be "self-explanatory"
 - The name explains the Business logic executed.
 - Input/Output are the logic information needed to do the job.
 - I have often made interfaces that "does too much"
 - Hard to use since input/output aren't self-explanatory
 - Hard to test and maintain because you influence a lot of "services"
-

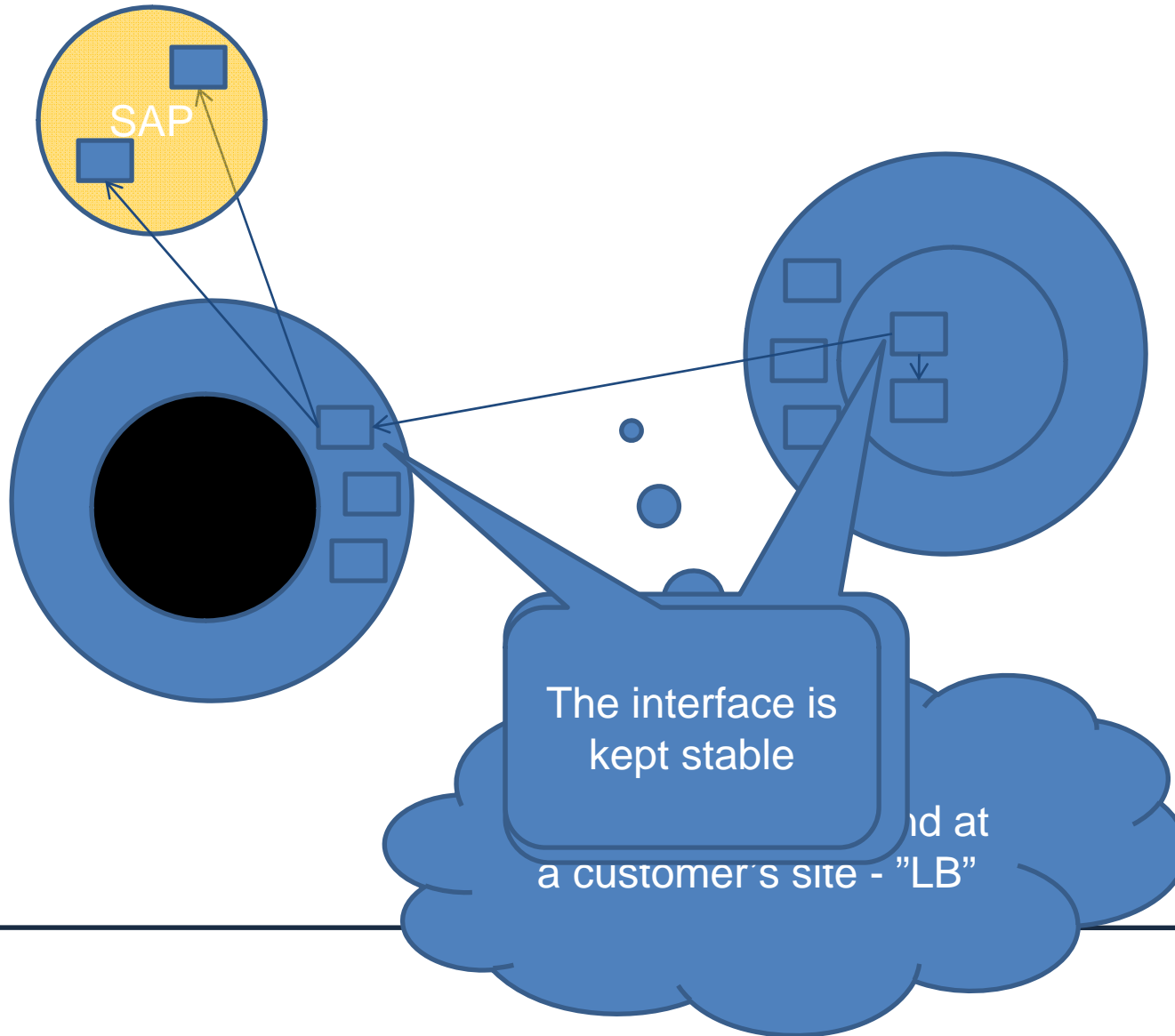
If we focus on the notion of webservices rather than architecture, SOA is great for:

- enabling communication between applications over the Internet
 - establishing communication between services in different systems
-

Loose coupling



Loose coupling



The interface is kept stable

and at a customer's site - "LB"

Demo

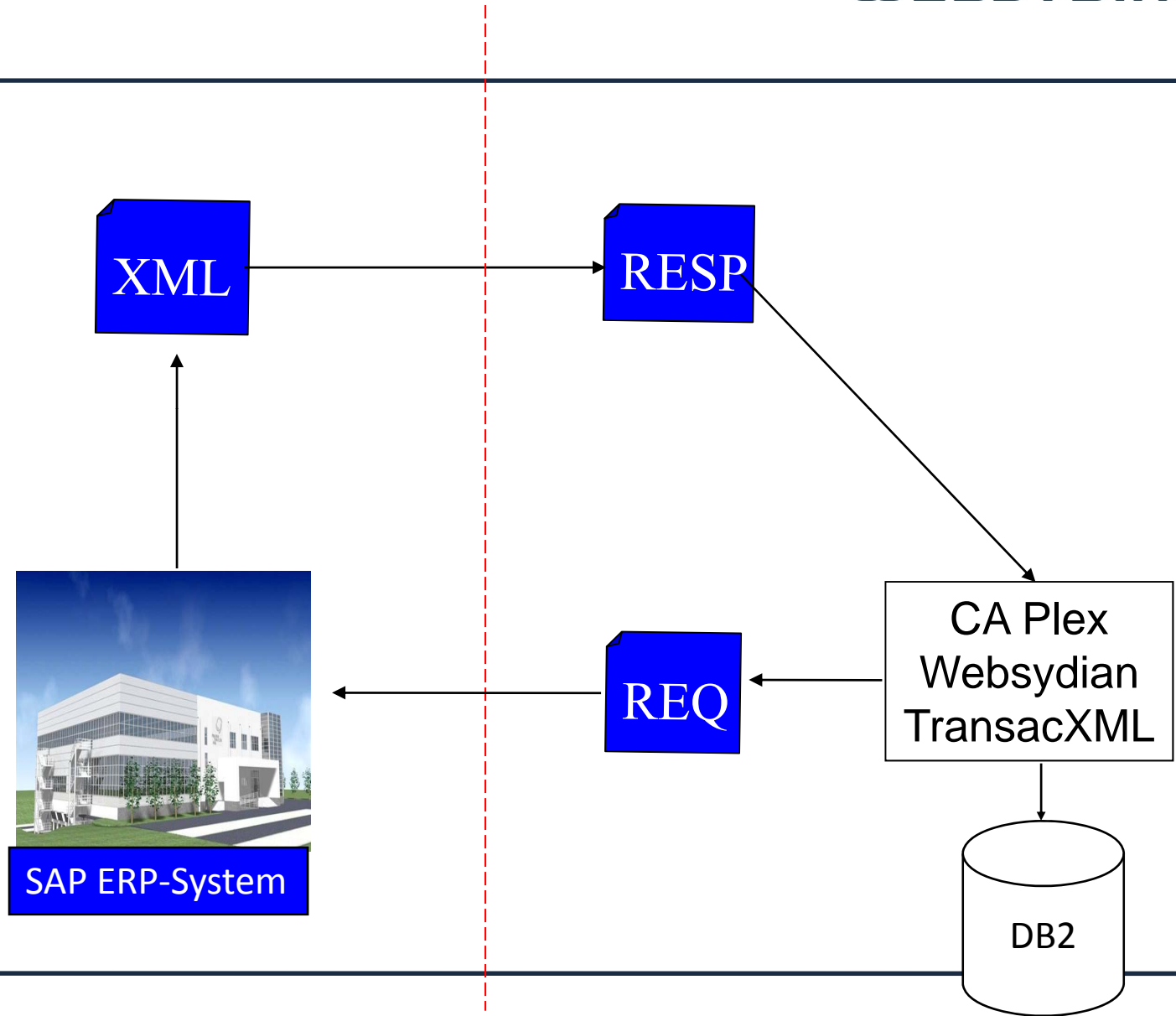
- How to develop the XML part of a Web Service using CA Plex and Websyidian TransacXML



-
- Management just bought a new ERP system.
 - The IT-department is required to keep periferal systems working
 - In this case the ERP-system is SAP.
 - The operation used in this demo is a Create Customer
 - Focus is on creating and receiving the XML – Not the transport layer HTTP, FTP, MAIL
-

Web Services – SAP Demo

WEBSYDIAN™



- How do I define the XML
 - How do I create the XML Document
 - How do I send the XML
 - How do I receive the XML Data Response
 - How do I traverse through the XML Sheets

 - Use TransacXML with WSDL/Schema Import
-

Live demo

Advice about SOA

- If it ain't broke – don't try to fix it
 - Reuse existing legacy systems
- Don't try to boil the ocean
 - Start out where it makes sense!!!

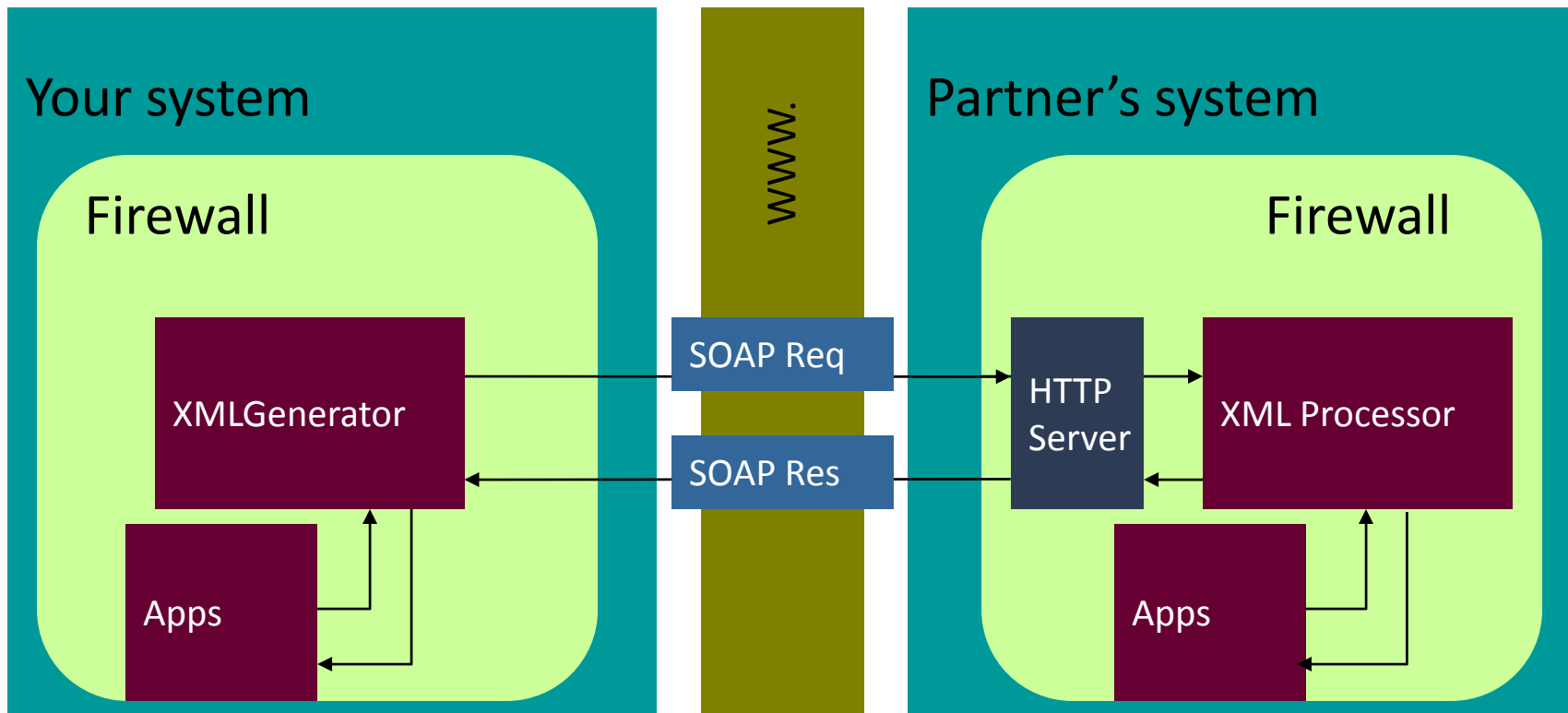


Another example

WEBSYDIAN™

- A partner requires that we interact with them using XML
 - In this case the partner has SAP
 - The operation used in this demo is a Create Customer
 - Focus is on creating and receiving the XML – Not the transport layer HTTP, FTP, MAIL
-

Document based Web Services



Q&A

Some new candy in WE 3.0

Webservice engine in WE 3.0

- Selector Program
 - File
 - FTP
 - Soap 1.1
 - Uwraps The XML
 - XML
 - R.E.S.T.
-

Some new candy in WE 3.0

- Criteria
 - SOAP1.1
 - SoapAction
 - XML
 - First element Name
 - X-Path
 - Request id (Soap action in Url)
 - URL (hardcoded)
 - HttpHeader
 - File
 - URL
-

Some new candy in WE 3.0

- XML processor = Your CA Plex Program
 - Open Architecture
 - You can make your own Selectors and Criterias based on abstractions in CA Plex.
-