Paradigm Shift for Agility through Web Services

Michael Stal
Senior Principal Engineer,
Center of Competence Middleware and Application Integration
Corporate Technology, CT SE 2
Phone: (089) 636 49380
mailto:Michael.Stal@mchp.siemens.de
http://hystrix.mchp.siemens.de/
The Problem

- Today, *anything* and *anyone* must be net-enabled.
- Automated business processes, products, and software systems need to evolve in „E-Time“.
- Everything must be changeable, extensible, adaptable.
- Quality is an important issue.
- *How can we balance these forces?*
In the stone ages (i.e., a few years ago) most applications were:
- non-distributed,
- running in an almost homogeneous environment,
- developed with a „stable“ set of requirements,
- using a limited set of external interfaces,
- using proprietary technology.

This approach does not work anymore.

In the new age (i.e., since the Web) most applications are:
- distributed,
- running in an increasingly heterogeneous environment,
- developed with an „instable“ set of requirements,
- interoperating with many different external interfaces,
- using standard technologies.

This approach works but has some drawbacks.
Why agility does matter:

- Developers cannot make too many assumptions about usage contexts and environments \textit{a priori}.
- Developers don't live on an island and must interoperate with legacy code.
- Running applications must be reconfigured instead of recompiled.
- Software Engineering must cope with change requests in Internet time.

Architectural consequences of these requirements:

- Software should not be designed as monolithic unit but partitioned into composable services that can be spontaneously connected and orchestrated by business/technical processes (\textit{component-based software}).
- „Software entropy“ should be maximized: loosely coupling between peers, decentralized information access, reflective approaches (\textit{Just-in-Time Integration}).
- Software must be \textit{e-enabled}.
- Legacy code must be integrated in order to protect investments.
All We Need is Middleware

- What we need is middleware that supports a Web of loosely coupled, composable, networked services. In detail:
  - We need **standard middleware** to connect applications, services, and legacy systems.
  - We need **XML applications** for data formatting and transfer between these software peers.
  - We need **Web protocols** for interoperability.
- However, these technologies are currently not well integrated with each other.
Web Services – Yet Another Paradigm Shift

- Web Services infrastructures integrate Web technologies, XML, and middleware. What we get is Web-based Middleware.
- With Web services the Web becomes the „Active Web“ where E-driven software processes and components interact with each other.
- Web Services enable ASP (Application Service Providing).
- All major companies have introduced their own proprietary Web Services platforms: IBM WebSphere, HP NetAction, Oracle DotNow, Microsoft DotNet, Sun ONE.
If Web Services are the solution, what is the problem?

- Information and services must be consumable from any device and from any place:
  - We need a platform that is device independent (virtual machine).

- New services must be composable from existing services and transparently accessible by consumers:
  - We need a middleware approach that provides code AND data interoperability (SOAP, UDDI, WSDL, XML).

- Support of „old-style“ Web content AND Web services is required:
  - We need advanced Web servers as gateways to Web pages and services.
What is the Problem (cont’d)

- Web services should be context-aware (security, preferences, transactions, location, …)
  - We need means to exchange and share contexts. And we need core services such as transaction monitors, database systems, calendars).

- Business processes and workflows should be automated:
  - We need workflow engines.

- Existing legacy code needs to be integrated:
  - We need connectors and standard middleware (J2EE, CORBA 3, COM+).
What is the Problem (cont’d)

- New flexible communication paradigms are needed:
  - We require Peer-to-Peer (P2P) and mobile solutions.

- Decentralized, heterogeneous systems are difficult to manage:
  - We need administration solutions.

- Non-functional requirements must be guaranteed:
  - We need support for fault-tolerance, load-balancing, multimedia streams.

- Last but not least, we need tools (*All you need is Code*):
  - We need programming environments, content management tools, ...
Core Elements of a Web Services Infrastructure

- **Workflow Engine**
- **Web-based and related Protocols (HTTP, XML, SMTP, ...)**
- **Service Discovery and Publication (UDDI)**
- **Service Description (WSDL)**
- **Service Context (Who, Where, When, Why, ....)**

**Web Service User/Provider**

- **Integration Layer**
  - Web Service Micro Service
  - Virtual Machine
  - Workflow Engine

- **Frontend Layer (Web Server)**
- **Backend Server**
- **Mainframe**
- **Legacy**

Agility Day
„Embrace Change“-Approach

- Out-of-the-box interoperability and integration through standardized communication protocols (XML, HTTP, ...).
- Dynamic discovery, integration, and binding of services (WSDL, UDDI).
- Service discovery and trading using standardized business interfaces.
- Orchestration of services using workflow engines.
- Advanced context-aware services.
- Integration of legacy code through standard middleware (EJB, COM+, CORBA 3).
- Device independance through virtual machines.
Sample Scenario

User
- login
- displayItems
- selectItem
- updatePage
- pay
- displayOrderInfo
- accept
- logoff

E-Shop
- create
- checkPreferences
- checkAvailability
- checkCredit

Inventory
- checkAvailability
- checkCredit

Credit Cart Company
- checkCredit
- sendOrder

Web Xpress Transport Co
- sendOrder

Events:
- CustomerObj
- updatePage
- pay
- accept
- logoff

URLs:
- www.e-shop.eu
- www.credit.eu
- www.transport.eu
.NET - The Microsoft Way of Life

.NET Foundation Services
- Passport, Calendar, Directory & Search, Notification & Messaging,
- Personalization, Web-Store/XML, Dynamic Delivery of Software and Services

.NET Framework & Tools
- ASP.NET
  - (Web Services, Web Forms, ASP.NET Application Services)
- Windows Forms
  - (Controls, Drawing, Windows Application Services)
- Base Classes
  - (ADO.NET, XML, Threading, IO, ....)
- Common Language Runtime
  - (Memory Management, Common Type System, Lifecycle Monitor)

.NET Servers
- SQL Server, Biztalk, Commerce, Exchange, Mobile Information,
- Host Integration, Application Center

.NET Devices
- TabletPC, PocketPC, ....
Here comes Sun ONE (Open Net Environment)
Comparison

- **Microsoft .NET**
  - Platform neutrality possible (Hailstorm)
  - Everything from the same vendor
  - Almost all parts already available, but most are not mature.

- **Sun ONE**
  - Java guarantees interoperability
  - Different vendors provide solutions
  - Not all parts available, but those which are very stable.
  - „Open“ standardisation (JCP)
Summary

- Web services are based on XML, middleware, and Web protocols.
- We are living in an agile world. Web services enable software agility by:
  - relying on standardized protocols and hiding other middleware details behind the Web server (loosely coupling),
  - using globally available services for discovery, description, and integration such as UDDI, WSDL (decentralized, reflective information),
  - supporting advanced (context-aware) services,
  - leveraging the Web browser for client-side integration,
  - (hopefully) introducing standardized domain-specific interfaces.
- Some problems still unresolved such as the handling of non-functional requirements.
- Interoperability between different technologies possible in theory (SOAP, UDDI, WSDL). In practice, vendor still matters.
- Don't forget to adapt your processes to the requirements of an E-Driven world. Visit the Agility Day on 17th, 18th July 😊
The End