MICROSOFT ÖSTERREICH

a view on

Microsoft Architectue Strategies in Context of current Trends

(Part I)

Mario Szpuszta

1/24/2007

Draft, v0.1

This document summarizes Mario Szpuszta's understanding of Microsoft's current thoughts and strategies on architecture trends such as Web 2.0, Software + Services and even classic trends such as Service Orientation, Business Process Management and Software Industrialization. The information provided in this paper is for information purposes, only. The material does not constitute the opinion of Microsoft and you should not rely on any material in this paper without seeking independent advice.

Table of Contents

Foreword, Introduction

I have started writing this two-part series of white papers for providing a summary of the presentation I am giving for the 18th Microsoft Architect Forum held at the Microsoft Office in Austria, Vienna. The primary intention of the whitepaper-series is giving Austria's architects information that summarizes my complete point of view on Microsoft's strategies in context of current architecture trends. Therefore the whitepaper-series is not supposed to cover details of each and every aspect of the strategies. It rather gives you an overall and complete picture on how the strategies fit together to a complete picture and refers to any details for each of the aspects. This first part focuses on the enterprise architecture strategies such as service orientation or business process management.

The overall story of the whitepaper-series and the presentation are both based on the Spark Meeting (1), a meeting of top IT architects held at last MIX (MIX06) in Las Vegas. The EDGE architecture discussed in this paper as a primary foundation was discussed during the Spark meeting (1) in Las Vegas and is part of the Spark model.

A Model in Context of Current Trends

Web 2.0 and Software +Services are current hypes and maybe trends which are in everyone's head. One of the first volunteers of Web 2.0 was Tim O'Reilly (2) who started working on the definition of what Web 2.0 is and which effects it will have to our current thinking. In his article on what Web 2.0 means (2) it turned out very quickly that Web 2.0 is much more than just about technology. "Harness collective intelligence" is a central aspect helped large web giants surviving Web 1.0(see beginning of (2)). Tim takes a look at typical examples such as Google, eBay or Amazon. Google's winning strategy for example was the page rank instead of classic search characteristics while the big advantages of eBay and Amazon against their classic competitors is the fact that they are out-reaching the number of customers compared to their competitors and especially the interaction they are able to do with users. For example Amazon has a huge number of reviews more per title than you can find anywhere else.

In each of these cases the "collective intelligence" was core to success – and it was driven by consumers which indicate a shift back to the consumer in software and IT while classic enterprise topics such as service orientation are necessary to support these trends. On the other hand the different attitudes and expectations between consumers and the enterprise definitely lead to more complexity which architects need to capture and understand. Exactly that was the target of the Spark Meeting in Las Vegas at (3). The first aspect discussed was the tree of life model (4) which is rather a business centric model putting consumer and supplier in relation to business and content (information). Without going too much into detail, one of the core consequences of the tree of life model was very similar to the model shown in Figure 1.

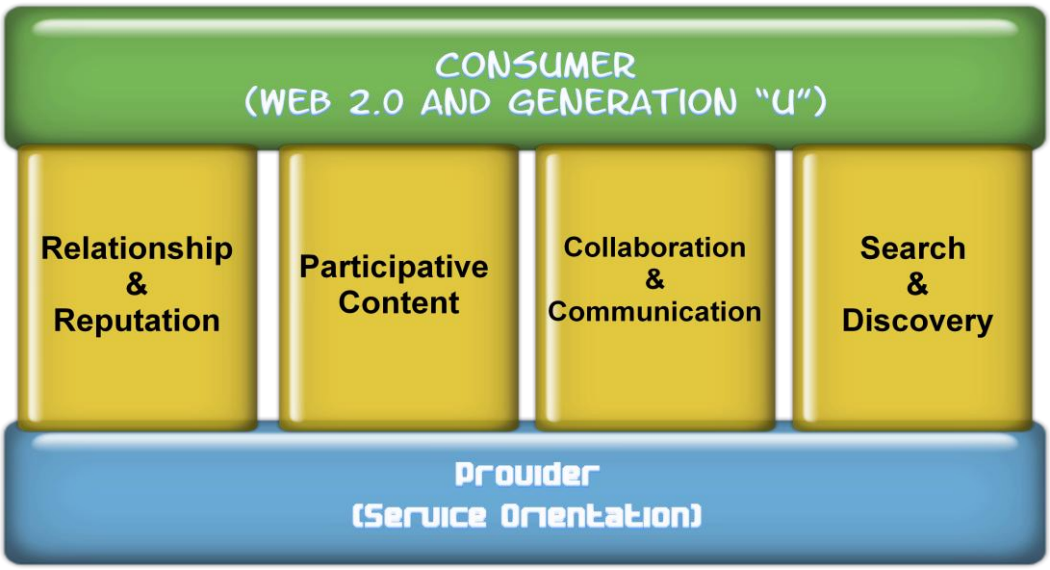

Figure 1: The EDGE Model

This model completes the enterprise focused models we usually have in mind with the consumer and the aspects which are influencing consumers and enterprises as well as aspects which are connecting these two worlds.

The primary differentiation between these two worlds is the fact that enterprises are typically interested in keeping their services and business as much under control as possible by relying on well defined infrastructures and standards. Of course the primary motivation for most organizations (except non-profit organizations) is generating revenue with their business. On the other hand the consumer's point of view is much different. Consumers are driven by subjective emotions and personal needs. They would like to find services, establish trust (more at an emotional rather than rational base) to services and consume them. Establishing a continuous relationship to consumers mean that enterprises need to give them the feeling they can control what an enterprise is doing to a certain extend.

Primarily these are aspects you can establish through a platform providing participative content. Typical examples are classics such as forums or ratings and newer examples are ratings, recensions as you know them from Amazon or eBay or much more blogs, WIKIs etc. On top of participative content it is important to allow your consumers working together on content which brings another level of complexity into the overall model: collaboration and communication where typical examples are again blogs and WIKIs but this time extended through instant messaging, discussion boards or even online conferencing.Last but not least finding information as well as individuals is the last core dimension being added to this model. Important to note is that consumers want to be found in addition to just finding information. Just think about all the famous bloggers which are trying to get onto the top-10 search results list of Live.com or Google to gain credibility and visibility out of their work and investigations. In many cases being referenced by others and therefore discovered easier is a key-driver of bloggers. Of course being discovered can influence a person's business dramatically as well (just think about you as an individual consultant or the CEO of a consulting company being discovered with many blog posts to your expert topic).

This model is the foundation of our discussion for the 18th architect forum and is the foundation for the remaining parts of the two papers as well. When reflecting this model in my opinion the enterprise-EDGE in the model summarizes anything we know from the trends observed in the past years – in particular service orientation and all aspects surrounding service orientation. On the other hand the consumer level really focuses on everything Web 2.0 is all about: collective power, participation and overall user experience. This first part of a series of two parts focuses on the enterprise-EDGE of the model introduced in Figure 1 while the second part will focus on the consumer-EDGE and Microsoft's architecture strategies in this space.

Enterprise View – Microsoft and Service Orientation

In the first part of the paper we will focus on the enterprise view of the model introduced in Figure 1. As enterprises focus on agility with their business processes, integration and interoperability and management with the ultimate goal to optimize their business and react as fast as possible to changes on the markets, service orientation and anything surrounding service orientation such as business process management, user experience in enterprises and governance (meaning management) is core to this part of the paper.

Three-Part View Model

In the past years Microsoft has established a three-part model for building system based on the service oriented paradigm as outlined in an article of the Microsoft Architecture Journal (5) from Beat Schwegler and Arvindra Sehmi on MSDN (6). In the three-part model as you can see in Figure 2, Microsoft outlines a clear strategy for designing service oriented systems. The model shows a clear path from your business through a service model (which outlines the architecture of your system) to a technical model that supports the creation of a service oriented solution.

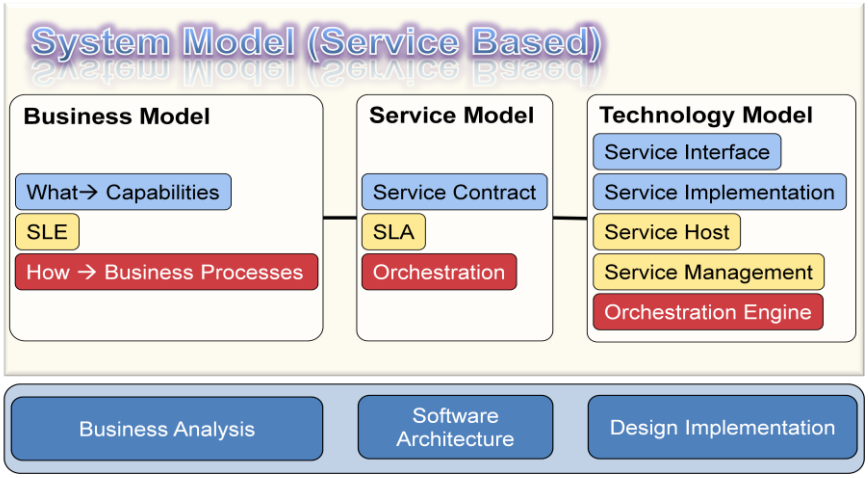

Figure 2: Three-Part Model on Service Orientation

Opposed to classic models which tried to go from business models directly to a technical implementation the service model fills a gap between the business model and the technology model with a service model. It furthermore adds another dimension within each of the models helping focus on the right level of granularity even within a model. Taking a look at the top-down approach you see a clear path from “what” needs to be provided to “how” it needs to be provided in each of the model. This shifts the focus from a pure process oriented view to a business centric view in terms of “what” the business wants and needs to supply. A focus on what a business needs to and wants to supply helps finding an appropriate granularity for your services in context of the business instead of making decisions based on the much more complex business processes, which are in fact an implementation detail in each of the models (from business through service to the technology model). Furthermore the model defines clear responsibilities for business analysts, solution/enterprise architects and application architects.

With this model you have a clear path for building service oriented systems in your hand. As an intermediary the service model decouples the business model from the technical model and therefore avoids that the technical model influences the business model. This adds an additional layer of flexibility to the model.

The Business Model – Capturing and Organizing Business Capabilities

The approach of creating business model in context of the system model introduced in Figure 2 is different to classic business modeling approaches. To create an effective business model, you need to be able to conceptualize the business and identify its core business functions (6). Basically a business function is a capacity or value a company can provide to the society. The methodology for identifying core business functions and bring them into a formalized model is called *capability mapping* as outlined in Beat's and Arvindra's article and in Ulrich Homann's article (7) in detail. An interesting part of business capabilities is the fact that they capture a much more stable view of the business compared to business processes which tend to change frequently (7). This is especially important when designing your services for a service oriented solution! Services are often used for exposing business functions to the outside world (anything outside a closed system). Of course business functions are exactly what the outside world is interested in whereas details such as process optimizations are relevant and interesting to the internals of a system.

Therefore capability-mapping starts with identifying the core business functions which means a clear, straight focus on "what" the business wants to deliver and provide. These abilities provided to the outside world of a system is the stable part of the business that does not (or should not) change frequently (if it does, the company should ask itself whether it knows what it wants to do and deliver and which value it provides to the society). Each business function identified is called business capability. Business capabilities are captured in a so called capability model as shown in Figure 3. Actually the model represented in Figure 3 applies to most of the typical companies which is important to keep in mind for Motion – a methodology referred to later in this paper.

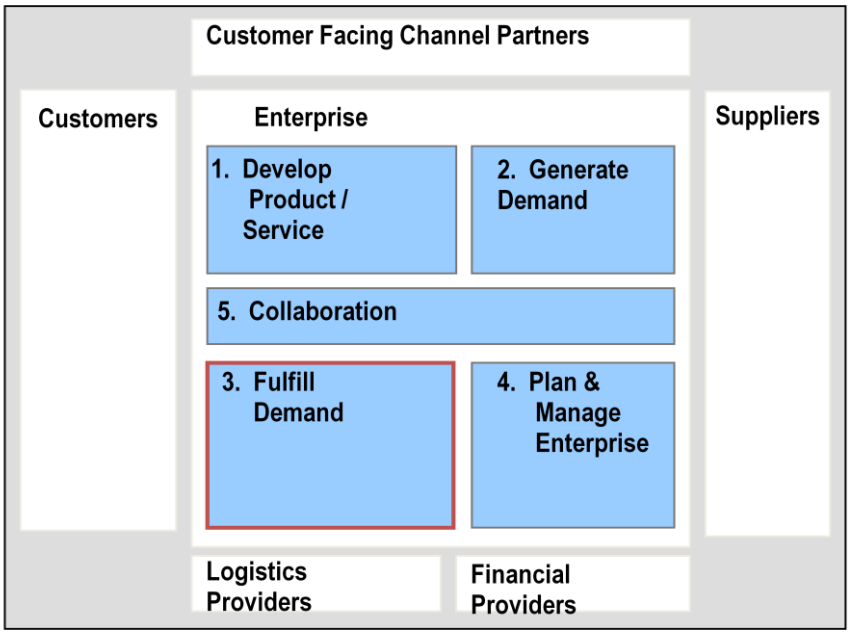

Figure 3: Level 1 Capability Map

As a business capability can be composed of other business capabilities to fulfill its purpose, a business capability model is a nested hierarchy of business capabilities (7). Having capabilities identified allows you specifying more attributes of each capability without going into further details on how exactly the capability is going to be implemented. If you know the value provided by a capability you can start thinking about the importance of what's being produced by a capability. This can be expressed through service level expectations (SLE). Furthermore you can design the interface of the capability which basically describes what a capability needs to have before it can provide any value and what exactly the outcome of the capability is as outlined in Figure 4. Also keep in mind that capabilities are a black-box, the only things you need to know as a consumer of a capability is its interface (what is required, what does it deliver) and its service level expectations and other parameters (e.g. such as billing).

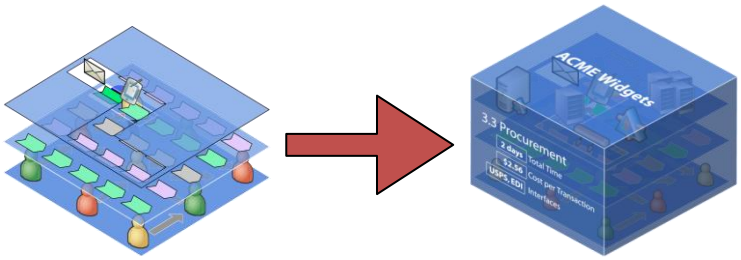

Figure 4: Capabilities are a black box

Next you start thinking about the implementation details of the capability which is basically analyzing the processes and people which are necessary for executing the business function expressed as capability. As shown in Figure 4 these are aspects captured inside a capability whereas from the consumer's perspective a capability is a black box. Within the process of specifying the details of a capability you can follow a similar top-down approach by start identifying, "what" is necessary to execute a business function, specify the details such as SLE on these capabilities and then think about the processes and the people required for executing the process within a capability. In another article, Ulrich Homann and Jon Tobey outline a complete process for developing capability maps within for phases which are basically captured in the previous part of this paper (8).

But the world is not as simple as outlined above. Typically business capabilities at a very high level span the entire value chain meaning that they are not delivered by just one company. Therefore business capability mapping introduces three levels of capabilities: foundation capabilities, capability groups and business capabilities (7)(6). While foundation capabilities address an entire ecosystem while capabilities group describe one part of an ecosystem and business capabilities are concrete building blocks for implementing business functions. You can find more details on these levels in Homanns article (7). As business capability modeling can get fairly complex, Microsoft has developed and refined a methodology called Motion and Motion Lite (9) which can be implemented in an organization together with Microsoft Services.

My personal opinion and experience has shown that even if you don't want to establish the whole business capability modeling process and/or Motion within your organization, business capabilities within the context of a specific application are a useful and effective way for providing a well-defined foundation for a service model with an appropriate level of granularity. For example, we have used use case diagrams as notation for modeling business capabilities in various projects in context of a specific enterprise solution. The advantage again is keeping the focus on stable aspects of the solution in relation to other enterprise solutions which are going to use the solution to be built. The approach we have used in several projects in Austria oulined looks as follows:

1. Identify the consumers of your solution. These can be modeled as actors within your use case diagram.
2. Identify business functions the consumers of your solutions expect from your solution. These business functions are the capabilities which you capture either in a business capability map introduced in this section or in use case diagrams. The only reason for using use cases is the broad availability of tools for creating use case diagrams on the market.
3. The next level of granularity is the activity diagram as you are used to from UML modeling. But the use case diagrams are now used in context of outlining the child-capabilities which are necessary for fulfilling the business function. There are basically two advantages of this approach: first it helps you figuring out whether another, previously identified capability is reused here (which finally means reuse of another service in the final solution) and identifying new, necessary capabilities.
4. The last step outlines the detailed process within a business function either through flow-charts or sequence diagrams. Within these sequence diagrams you can get into details of the overall architecture of the components orchestrated within the business function. By drawing a line between the consumer and the components/activities hidden behind the business function you can identify the service façade including the messages required to be sent to and returned from the business function.

This is some sort of a "pragmatic" approach for usage of business capabilities that allows you to use existing modeling tools for capturing the business models by still remaining the level of granularity at an appropriate level for your service model.

From Business Capabilities to Services

Having your business capabilities including service level expectations in place allows you moving on to the next step within the process – defining your services, service contracts and service level agreements. Uwe Homann and Jon Tobey outline the process of getting from capability models to services in great detail based on Motion (8).

From my perspective there are two interesting abilities you get out from business capability maps: first an organization can decide on which business functions they want to and are able to deliver themselves and which business functions need to be outsourced to another organization. Second based on this decisions a capability model allows you identifying services where explicit handling of contract according to the four tenets (6) is necessary.

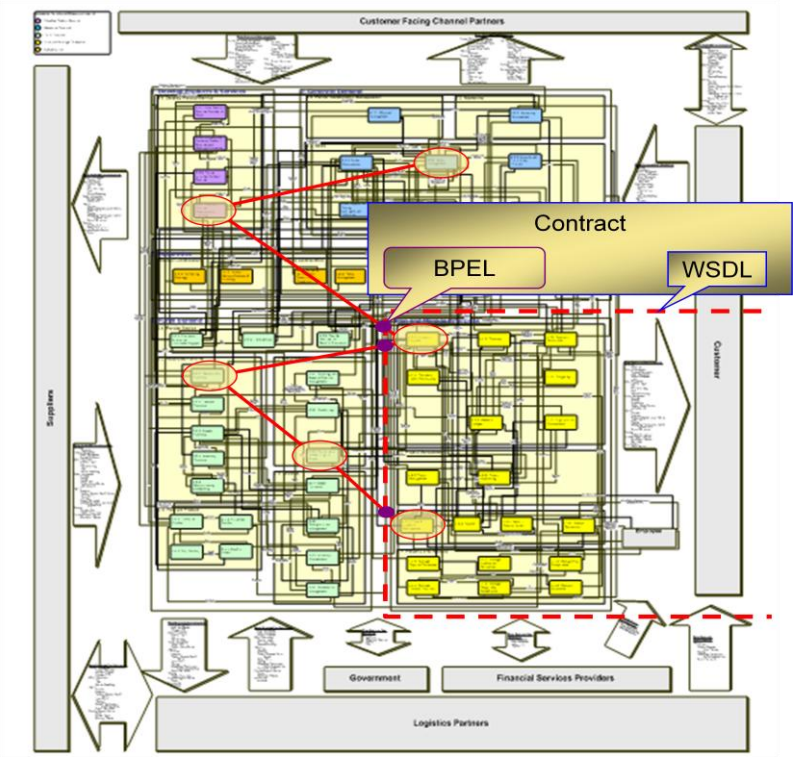

Figure 5: Identifying boundaries in your capability model

Figure 5 shows a sample capability model provided by Beat and Arvindra during one of their presentations at TechEd Europe 2005 in the architect pre conference. A red line outlines the decision of an organization on which capabilities it can provide itself and which one needs to be outsourced. An overall process spans capabilities delivered by the organization itself and other organizations. This red line differentiates the company's capabilities from the other capabilities and shows exactly where explicit management of contracts and interoperable technologies is especially important. In my opinion these are the places where the four tenets are playing a special role – within those borders optimizations and more pragmatic approaches are valid for each single service (not across services) provided at these borders as within those borders we are talking about autonomous implementations of services.

The Service Model

Business capability models deliver a foundation for deciding which services you are going to provide and what the service level expectations for these services are. Core of the service model is capturing entities, messages and service interfaces (6). It is especially important to capture these artifacts at a technology-independent level. Transport-protocols and implementation-artifacts such as message classes are finally captured by the technology model. Service interfaces of course should be designed based on the four tenets as outlined in several articles (10):

- Boundaries are explicit
- Services are autonomous
- Services share schema and contract
- Services share policies

These four tenets especially focus on lose coupling between service consumers and service providers. First of all as usually services are exposed across organizations within or even outside a company we are talking about explicit boundaries. That means you need to treat your service contracts explicitly and design them with versioning and extensibility in mind (11). Furthermore you may not pass internal business objects across these explicit boundaries as they might be optimized and changed over time which may not affect service consumers. Within a service boundary you can use classic approaches such as a layered architecture as summarized in Figure 6.

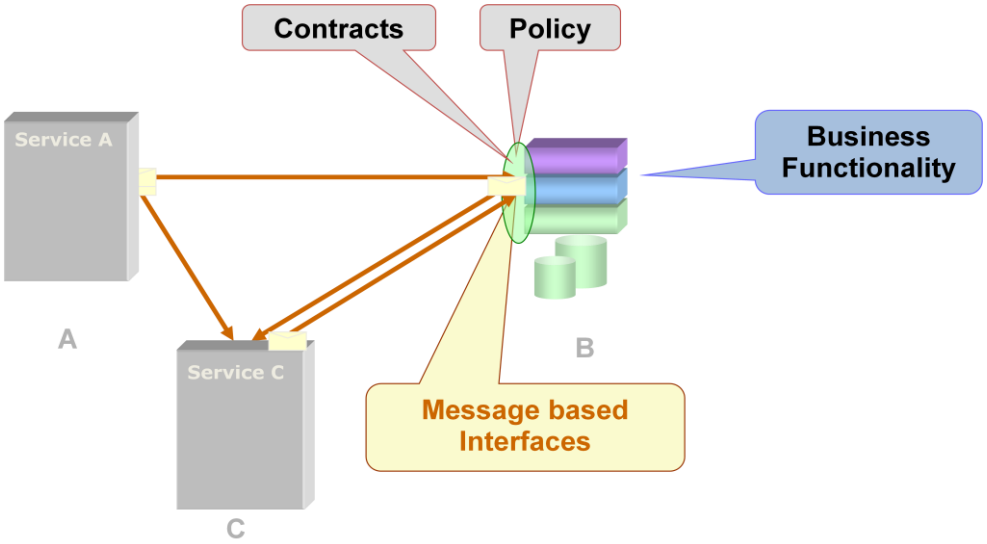

Figure 6: Services share contract and policy

The only difference within the layered architecture is the addition of agents for consuming other services and service facades. Both are mechanisms for decoupling from other endpoints and are providing mapping mechanisms from the internals of a system to the outside. Practically this results in a mapping between internal business objects and external messages. Whenever an internal implementation gets changed, these mapping layers need to be adopted without having any effects on the outside world.

Independent of the internal design of a service you can categorize services in your portfolio in a layered approach as well as shown in Figure 7 from a presentation of Harald Leitenmüller and Andreas Erlacher (12).

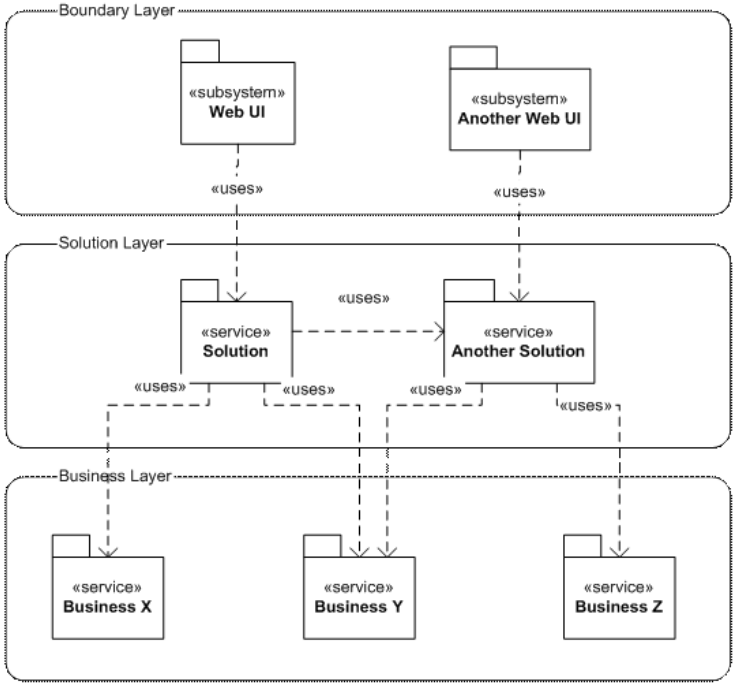

Figure 7: Layers of Services

The boundary layer implements a custom interface for a solution defined in the solution layer. The interfaces defined within this layer are used for providing access to a set of services from exactly one solution layer to the user. The big difference to a classic three-tier model is the fact that user-related services within the boundary layer are more than just a presentation layer on top of one business layer. Typically a solution will establish more than one service across different departments altogether providing information to the user. In most cases this information needs to be prepared for becoming useful to the user. Preparing information can go from aggregating information, providing effective ways for analyzing information or presenting information in a unified, modularized and configurable way across services. That means opposed to a classic presentation layer the boundary layer might access several autonomous services from one solution whereas each of these services is implemented through an n-tier layered approach themselves. Unlike the other layers the boundary layer generally does not consist of services in sense of the service oriented world.

A solution layer combines a set of services which are necessary for implementing higher-level business functions. A solution layer includes services in terms of service orientation which can be consumed by each type of consumer – the boundary layer, another solution or even another organization. Opposed to the boundary layer the solution layer may call services from another solution layer as well and therefore building a so called composite application (13). Business processes are encapsulated at the solution-layer level meaning that services at the solution layer level are orchestrating other types of services.

The business layer includes autonomous services which are encapsulating single business activities. These services can be re-used between solutions as the only difference to the solution layer is that these services encapsulate single business rules, activities and access to business entities, only.

Technical Model – Microsoft's Platform for building Services

For implementing a service model you need a technical model. Microsoft's core component for implementing services is Windows Communication Foundation (14). Its architecture was designed with service orientation in mind.

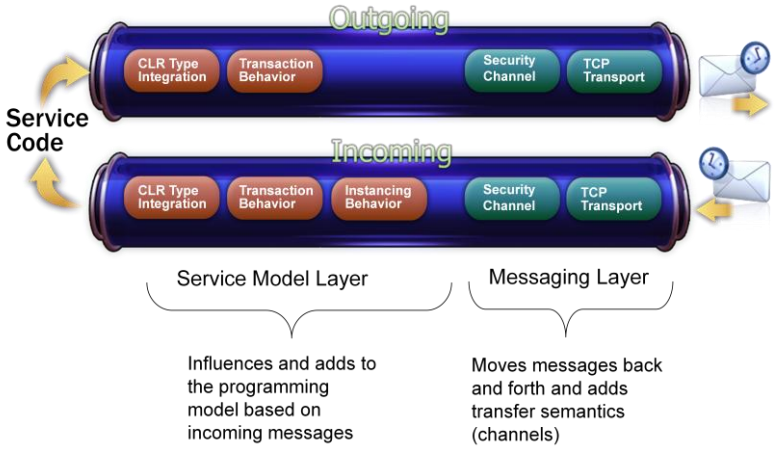

Figure 8: Windows Communication Foundation Architecture

As outlined in Figure 8, its design clearly differentiates between a messaging layer and the service model layer. The transport layer takes care of the transport details such as transport protocol and transport-level security while the service model layer takes care of higher-level, transport-independent aspects of the messaging. These aspects are transactions, message-level security such as security provided by WS-* specifications (15). Your service code is completely independent of any transport and messaging details. Therefore for example switching from one protocol like HTTP to another protocol such as TCP or vice versa is a matter of configuration as long as you don't change the message exchange pattern (asynchronous to synchronous or vice versa).

Furthermore WCF supports the four tenets by making a clear distinction between entities, messages and service interfaces as outlined in the architectural overview (16) and in the contract-design how-to article on MSDN (17). Beat Schwegler and Arvindra Sehmi outline in the second part of their article on Service Oriented Modeling (18), how-to get from a service model to a technology model using Windows Communication Foundation. More aspects of the technology model such as Windows Presentation Foundation (19), Windows Workflow Foundation (20) or for example BizTalk Server (21) and Office Business Applications (22) are outlined while discussing more concepts in the next sessions.

Business Process Management and Microsoft

As some services need to be orchestrated by other services to fulfill a higher-level (composite) business function in your business capability, this is the right point to address business process management (23) and Microsoft's view on BPM.

Microsoft's view on business process management is influenced by several architecture and market trends at the moment. Trends influencing this view are:

- Service Orientation as outlined previously
- Software Factories and industrialization of IT (24)
- User experience and consumer-oriented trends such as Web 2.0 (2)

As processes are a core part of services in a solution based on service oriented paradigms the influence service orientation on business process management is quite obvious. In my opinion, business process management will be heavily influenced by business capability modeling as orchestrating capabilities are a core part. Just the approach is a different one – while classic BMP modeling strategies put the process into the center, business capability modeling primarily focuses on the actual business functions at first and adds the process-aspect afterwards.

Software Factories and Domain Specific Languages (24) are definitely influencing Microsoft's strategy on business process management. As outlined by David Green in his article on Windows Workflow Foundation (20) in the

Architecture Journal (25) and in his interview on the Microsoft Austria Podcast Network (26) there is great potential in building domain specific workflow languages based on Windows Workflow Foundation. The primary intention is very similar to the one of Software Factories, just in context of Workflows. Instead of inventing one abstract language for all types of business processes Windows Workflow Foundation enables you the creation of a Workflow language that exactly meets the needs of your business domain through creation of custom activities (27). Rather than trying to get the business people used to an abstract model this approach bridges the gap between business and technology by bringing the technology closer to the business instead the other around. This approach has been implemented successfully, already, in a number of customer projects such as the order-process automation project of ONE GmbH. (28).

Another factor influencing Microsoft's strategy on business process management is the consumer-focus prefaced by the Web 2.0 trend. As user experience (29) includes anything people experience includes, it has an effect on services, business processes and of course emerging patterns and approaches in UI space.

Therefore Microsoft differentiates between several levels of business processes architects need to take into account as shown in Figure 9.

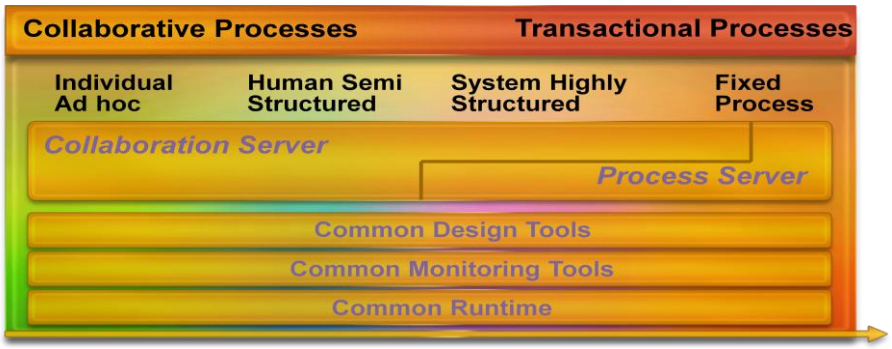

Figure 9: Levels of Business Processes

Processes which are orchestrating services within an automated business process are on the "fixed process" edge of the continuum of processes. But many processes within companies are executed on a very ad hoc, individual fashion through internal web portals, via email or even instant messaging. These types of processes are captured on the other end of the continuum shown in Figure 9. In the end, a platform, a solution and a solution-architecture needs to support both ends of the story while still keeping an appropriate and required level of control and discipline within an organization. To formalize ad hoc processes and make sure that they really get executed in time, an infrastructure is required for quickly setting up those processes and helping individuals getting the process done. This work usually needs to be done by a collaboration infrastructure such as Windows SharePoint Services (30) in conjunction with Windows Workflow Foundation (20) and an easy way for creating such workflows such as SharePoint Designer. The collaboration platform can build a bridge to the services encapsulating business processes through a process server such as BizTalk Server (21) on the Microsoft platform by providing mechanisms for attaching metadata to artifacts managed and processed through the collaboration platform. Examples are SharePoint Content Types (31) or the possibility of attaching schemas to the new Microsoft Office Open XML File Formats (32). Such mechanisms allow the creation of semi structured and highly structured information that can be processed by services orchestrated within a backend process server.

Furthermore Microsoft's strategy with Windows Workflow Foundation (20) is providing a single workflow platform for any type of application – reaching from collaboration products and process servers outlined above to custom workflow applications and user interface processes for controlling navigation within the boundary layer (previous section of this paper).

Microsoft's Strategy on Enterprise Service Bus

While service orientation as a paradigm provides guidelines on identifying services and designing services according to the four tenets, it does not make any statement on organizing services with the goal of keeping the service landscape of an enterprise under control. This is where the concept of an Enterprise Service Bus (33) gets into the game as a key component of a service oriented infrastructure yet many people have a different understanding on ESB. Taking a look at common ESB products, in an article back in 2004 (34) and a recent presentation on guidance for ESB (35) Microsoft has gathered together the following commonalities of almost all ESB architectures as follows:

- Brokered communication
- Address indirection and intelligent message routing
- Web services support
- Endpoint metadata
- *Transformation*
- *Orchestration*
- *Adaptation*

As a result, last Winter Microsoft announced that it will provide architectural guidance (35) based on a real-world project including documentation and a ready-to use framework including source code (but without support) for building an ESB infrastructure during calendar year 2007. The overall architecture of this ESB infrastructure is outlined in Figure 10.

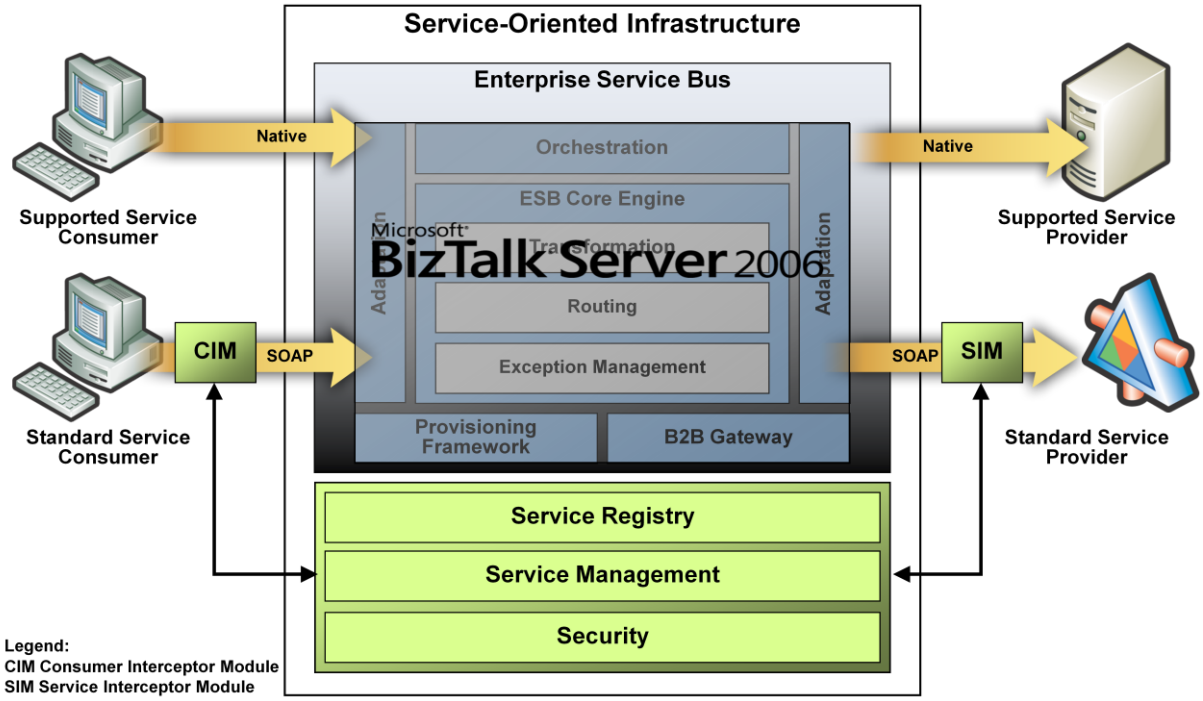

Figure 10: Enterprise Service Bus architecture guidance

This guidance will be based on BizTalk Server which provides the necessary foundation for orchestration, transformation and adaptation. On top of BizTalk server the guidance will include concepts and reference-implementations for building a service registry and managing services as well as endpoint data to enable intelligent message routing and brokered communication. This guidance will be based on the .NET Framework 3.0 and its components. Together with Business Activity Monitoring and tracing capabilities of BizTalk the guidance will include a reporting and monitoring infrastructure together with SharePoint as a front-facing infrastructure. But BizTalk and .NET are not the only way for building an ESB infrastructure with the Microsoft platform. Depending on your environment you can use other technologies such as SQL Server Service broker for building a data-driven ESB infrastructure which is especially interesting for a set of entity services implemented through SQL Server 2005 functionality such as stored procedures exposed as entity services through an HTTP endpoint. But in the end this will be appropriate for very specific scenarios, only!

User Experience for Architects – Composition is Core

According to its common definition (29), user experience incorporates much more than just the user interface itself. Finally it includes anything people experience includes. People experience is influenced by response time of the UI, overall performance of the application, usability of the user interface, general emotions in terms of the user interface and its appearance, usefulness of the tasks and of the information provided by the application. User experience therefore influences several parts of the user interface and the way the frontend is consuming services either directly or indirectly. These factors are summarized in Figure 11 (by Szpuszta).

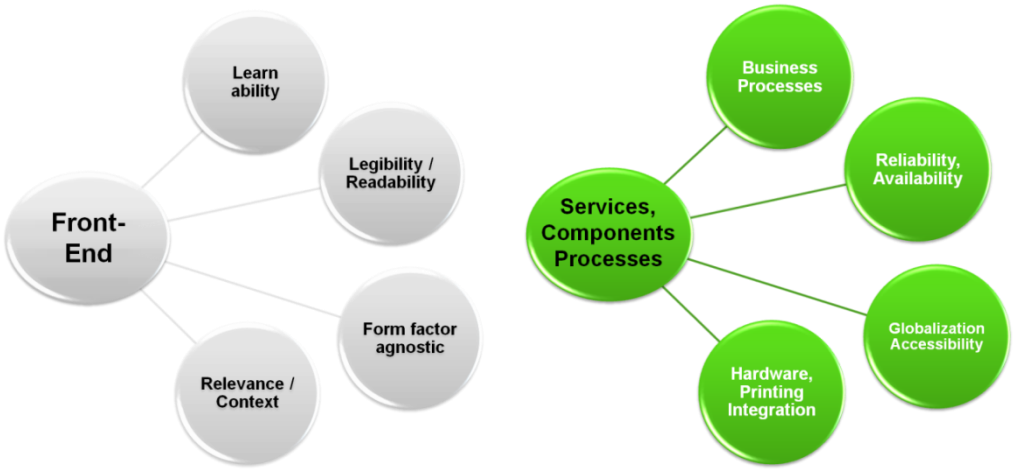

Figure 11: A complete picture on user experience (Szpuszta)

Especially the usefulness of information provided by the application combined with the overall performance and response time as effects on how clients are accessing services and if they are accessing services directly at all. In my opinion, creating useful information out of a service portfolio which has been built with interoperability, integration and automated business as core requirements can get a very hard challenge. In addition information processed and supplied by services created for transactional business is usually not optimized for analysis and decision support systems. Therefore although not including business logic in terms of business rules and activities the boundary might include some additional logic for aggregating, preparing and caching information optimized for the core requirements of the user. This is much like an integration- and caching-layer on top of the services provided by a solution to optimize communication with the end user as shown in Figure 12 from an article of Gianpaolo Carraro on Software as a Service (36).

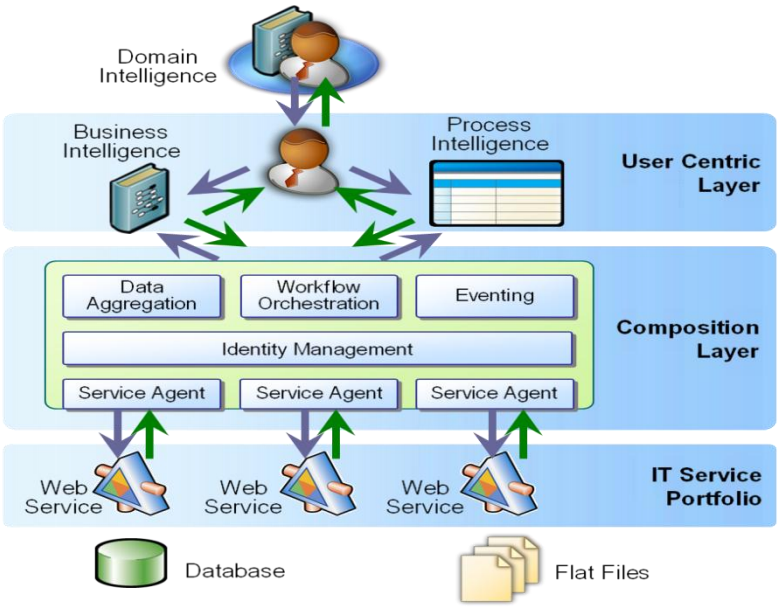

Figure 12: Composite Applications

There are different, independent types of core components of user-centric composite applications. Composite applications can be service based including business intelligence with a presentation layer on top for delivering the information to the end users on a dashboard, a portal or within a client application. With the 2007 Microsoft Office System and SQL Server 2005 Microsoft provides an infrastructure for building this type of composite applications as outlined in an article on MSDN from last December (37). These types of applications are specific flavors of so called Office Business Applications (22). 2007 Microsoft Office system delivers a comprehensive platform on the server and on the client for creating composite applications.

Another breed of composite applications appeared within the last years are composite smart clients (38). These are special types of extensible smart clients which are extensible by dynamically loading modules based on a configuration and the user's security context into a common shell. The advantage of composite smart clients primarily is providing a standard user experience by delivering a common shell across all applications because the applications are loaded as modules into the shell. At the same time composite smart clients allow reuse of common, client-side services (such as services encapsulating offline capabilities) across modules and therefore reducing maintenance and management costs. Microsoft delivers a framework for building composite smart clients based on the .NET Framework called Composite UI Application block (39). In addition Microsoft provides a set of automated developer guides, reference implementations and a huge set of documentation on using the Composite UI application block with the Smart Client Software Factory (40).

There are a number of case studies and projects using CAB and SCSF in Austria out there, already such as the Test Management Factory System of AVL List (41). Another example and one of the largest projects based on Composite UI application block and Smart Client Software Factory is the common bank desktop developed by RACON Software GmbH., a company of the Raiffeisen Group in Austria. During this project I have developed a whitepaper providing architectural guidance on building composite smart clients using CAB/SCSF with clear approaches on identifying, designing and packaging artifacts based on a use case driven analysis. This whitepaper has been published on microsoft.com (42) and the architectural part with a more detailed description on the scenarios and motivations is part of issue 10 of the Microsoft Architecture Journal (5).

Business processes are playing an important role in terms of user experience as well. The reason for this is outlined in the section "

Technical Model – Microsoft's Platform for building Services

For implementing a service model you need a technical model. Microsoft's core component for implementing services is Windows Communication Foundation . Its architecture was designed with service orientation in mind.

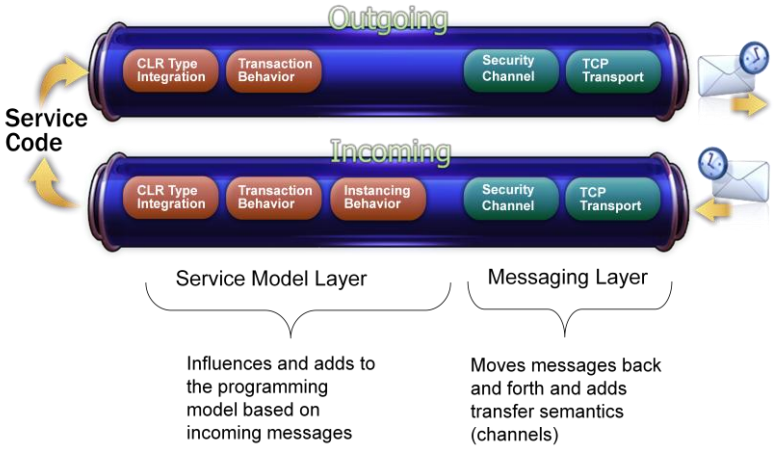

Figure 8: Windows Communication Foundation Architecture

As outlined in Figure 8, its design clearly differentiates between a messaging layer and the service model layer. The transport layer takes care of the transport details such as transport protocol and transport-level security while the service model layer takes care of higher-level, transport-independent aspects of the messaging. These aspects are transactions, message-level security such as security provided by WS-* specifications . Your service code is completely independent of any transport and messaging details. Therefore for example switching from one protocol like HTTP to another protocol such as TCP or vice versa is a matter of configuration as long as you don't change the message exchange pattern (asynchronous to synchronous or vice versa).

Furthermore WCF supports the four tenets by making a clear distinction between entities, messages and service interfaces as outlined in the architectural overview and in the contract-design how-to article on MSDN . Beat Schwegler and Arvindra Sehmi outline in the second part of their article on Service Oriented Modeling , how-to get from a service model to a technology model using Windows Communication Foundation. More aspects of the technology model such as Windows Presentation Foundation , Windows Workflow Foundation or for example BizTalk Server and Office Business Applications are outlined while discussing more concepts in the next sessions.

Business Process Management and Microsoft" earlier in this whitepaper where I mention the distinction between ad hoc individual processes and fixed processes. Individual, ad-hoc processes are mechanisms for providing a better user experience by allowing users to define certain, well-defined parts of an overall process as it fits best to them. In terms of communication between the user (boundary layer) and fixed back-end processes it might be necessary to expose a composition-service layer optimized on delivering information from a process to the user in a useful, aggregated fashion whenever the process requires the user to deliver information. Figure 13 outlines a Unit-of-Work pattern demonstrating this concept.

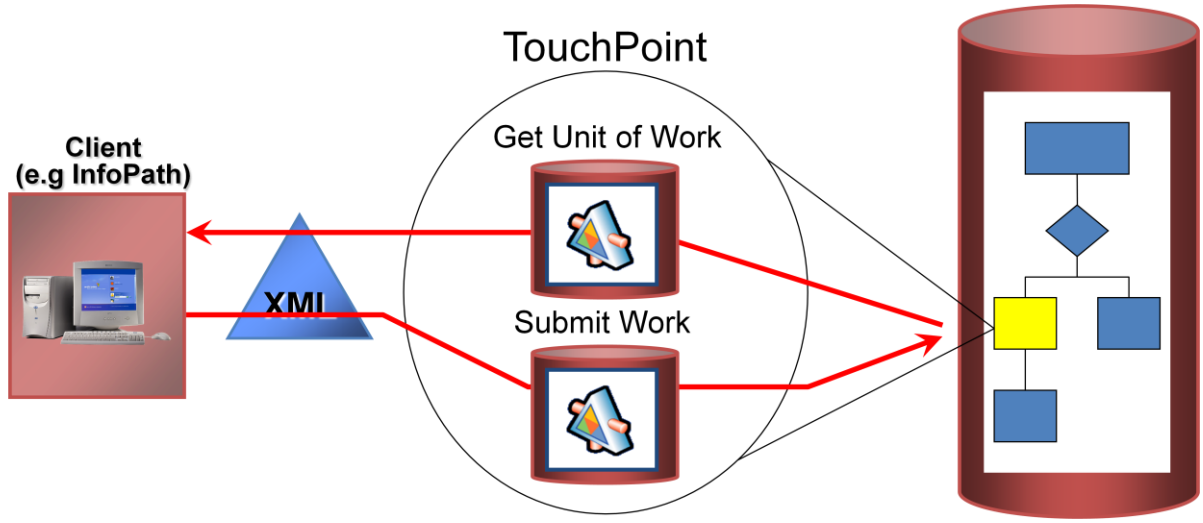

Figure 13: Unit of Work and TouchPoint Pattern

The touch-point is basically a service-composition layer at the boundary level gathering all the required information for the user from services of the solution layer the user requires to put together the information for a process to continue. Whenever the user submits the data back to the process layer through a service, the composition layer (touch-point) takes the information and forwards it to the one or several services on the process layer so that the backend process is able to continue doing its work. This is especially a pattern that perfectly works for Microsoft Office InfoPath-based solutions where data for filling out forms can be retrieved through data connections and can be submitted to web services.

Finally user experience is definitely influenced by the look & feel and the appearance of the user interface. If users don't like the user interfaces your applications are providing, they will just not use these applications. Especially with Web 2.0 (2) and its focus on consumers user experience gets more and more important – for both, web applications and smart clients. That brings up the first question: when create a smart client and when create a web client? Of course the answer is: depends on your requirements. As Beat mentioned in one of his first presentations on smart clients back in 2003, the question you need to ask yourself needs to be asked with the following guideline in your mind: "It's not about smart versus thin; it's about reach versus rich".

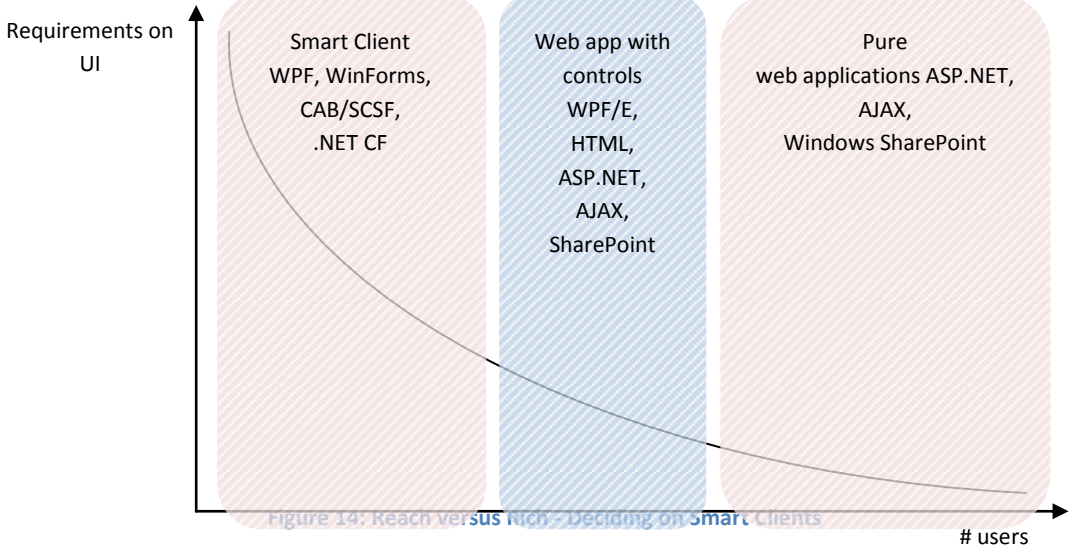

Figure 14: Reach versus Rich - Deciding on Smart Clients

As outlined in Figure 14 with an increasing number of users classic web technologies such as HTML, ASP.NET and AJAX are the way to go whereas within increasing requirements on the user interface smart client technologies such as Windows Presentation Foundation (19) are more appropriate (typically with increasing requirements on the UI the number of users gets smaller decreasing the effort for managing smart clients).

If you have decided for an approach on delivering user experience to your end users, you need to make sure that your UI will be accepted. Especially for larger, more complex applications this involves special user interface designers which are responsible for usability and an appropriate design of the user interface whereas developers should stay with implementing the functionality behind the UI. While this has been a common practice for web applications for some time now, lack of support of tools made this nearly impossible for Windows smart client application development. Windows Presentation Foundation (19) addresses this issue by providing a declarative language for creating user interfaces with the XML-based eXtensible Application Markup Language (43). In reality XAML is a declarative, XML-based language for creating and initializing any tree of objects. WPF uses XAML to create and initialize a tree of user interface objects, but at the same time WF (44) uses XAML to create and initialize a tree of activities within a workflow.

Getting back to user interfaces and WPF, XAML enables a clear differentiation between user interface designers and developers as outlined in

Figure 15: XAML, the designer and developer

With the Expression-tools (45) Microsoft provides tools for professional designers. These tools allow designers either creating Visual Studio projects from scratch for designing professional user interfaces and interactions or just take existing Visual Studio projects and modify user interface designs previously created (or modified) by developers. Expression Blend will support design of Windows Presentation Foundation based user interfaces while Expression Web enables professional design of ASP.NET based web pages.

Model-based management

Last but not least you need to manage your IT service portfolio. In general Microsoft provides ITIL-based organizational management guidelines through its Microsoft Operations Framework (46). Microsoft's primary management infrastructure is Microsoft Operations Manager. For web services based management, Microsoft announced a strategic partnership with Amberpoint (47) back in 2003 for integrating their web services based management platform with Microsoft's offerings. With all these concepts and guidelines offered by the Microsoft Patterns & Practices Group (48) you have a complete infrastructure for managing your IT service portfolio in your hand, today.

But the long-term vision of management is a model-based management strategy currently produced by the Dynamic Systems Initiative (49). Primarily the idea is having a complete, tool supported integration between software architects and IT architects with the ultimate target of catching possible problems in the application architecture or the infrastructure architecture that will cause deployment to fail as early as possible and in an ongoing and up2date fashion.

As outlined previously in this whitepaper, every role within the development process of a service oriented solution works on certain types of models. The difference is just the level of abstraction. The business analyst works on business models, the software architect works on service models and application models while the IT architect finally works on infrastructure models. As everyone should specify requirements for the artifacts in the models, by mapping the artifacts of one model to another model (business model to service model and service model to infrastructure model at a very high level of abstractions), these models can be validated against each other in order to figure out possible deployment show-stoppers as early as possible. This process is demonstrated in Figure 16 with what's possible in Visual Studio Team System today.

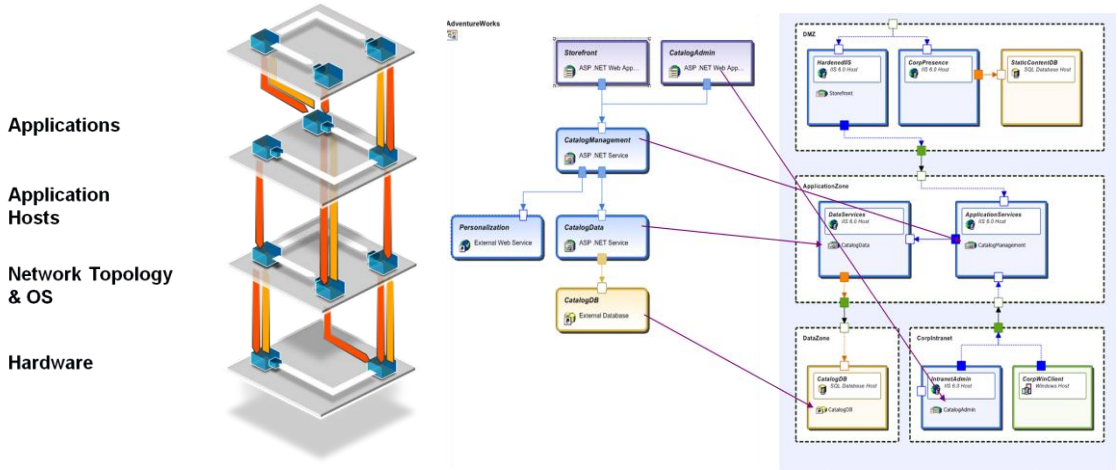

Figure 16: Mapping Models

But as long as the models are just models without any connection to reality they will get out-dated soon. This is where the Dynamic Systems Initiative (49) gets really interesting. One of the primary targets of DSI is defining a standard called Service Modeling Language (50). Service modeling language is based on XML and defines a way creating a formal model for a complete system with all the artifacts contained in this model. As soon as an operation infrastructure product such as System Center (51) on the Microsoft platform understand service modeling language it can compare the reality with the model through its infrastructure inventory services for example. This allows you keeping infrastructure models up2date all the time. On the other end, Visual Studio is able to keep application architecture and system models up2date and in sync with code developers are writing, today, already. Finally this approach allows you an ongoing end-to-end validation of your models all the time with the advantage of having models reflecting the real situation. SML support will be added to Windows Server and System Center during the Longhorn Server time frame in calendar year 2007/2008.

Software Factories as an Approach for Building Systems

The concept of Software Factories was introduced the first time back in 2004 by Jack Greenfield, Keith Short, Steve Cook, Stuart Kent and John Crupi. Software Factories is all about industrialization of software development. Software factories are influenced by other industries such as the car industry where production happens with high-performance, optimized production streets in a highly scalable fashion. It is supposed to change the way we are developing software based on patterns, languages and tools.

A software factory is a software product line that provides a production facility for the product family by configuring extensible tools using a software template based on a software schema (52).

One of the primary assumptions of software factories is that the perfect language for solving every problem has not been invented yet, and will never be invented. Therefore instead of investigating in more generic, abstract languages and solving problems with these types of languages we should investigate in languages optimized to a very specific family of applications (such as CRM systems) which can be used by architects or even business people to create (model) applications based on this language. Such languages are called Domain Specific Languages (53). These languages are typically based on a framework for specific types of applications and are used for putting bits and pieces together by designs parts of a system left open by the framework. The result of the final design is an artifact or a set of artifacts that completes the existing framework according to the design the architect/developer has created for building up the complete solution. If you want to learn more about software factories I'd really recommend reading Jack Greenfield's and Keith Short's book *Software Factories* (54).

What we have Today

Right know we have two maturity-levels of Software Factories: real software factories using domain specific languages and software factories with automated developer tasks integrated into Visual Studio. The second software factories are provided by the Patterns & Practices Group (48) and are based on a component called Guidance Automation Toolkit and Guidance Automation Extensions (55). Primarily the GAX/GAT approach is built on a three-phase model as outlined in

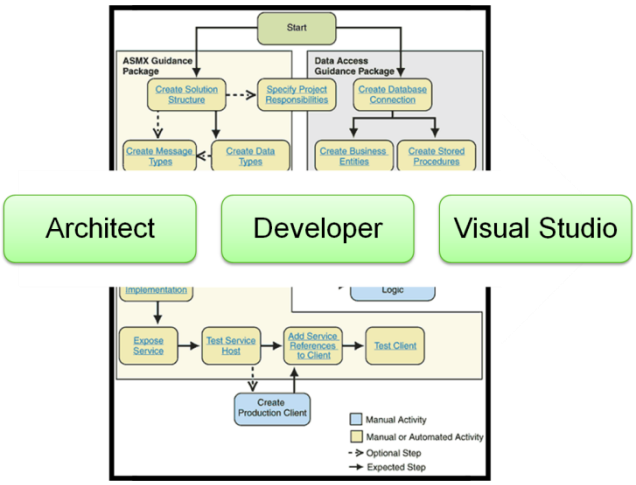

Figure 17: Guidance Automation Extensions

First of all the architect defines guidance on how-to use specific components or patterns of a custom framework or a general-purpose framework. To formalize tool-driven support, together with the lead developer the architect creates guidance packages which are primarily responsible for generating code based on receipts executed and parameterized by the line-of-business developers of your development team. These guidance packages can then be deployed to Visual Studio 2005 so that line-of-business developers can use them. Guidance packages are typically launched from within Visual Studio either through an additional window added to the IDE workspace by the Guidance Automation Extensions called Guidance Explorer or through context menus which are displayed in certain situations whereas the architect can define these situations exactly when creating guidance packages.

Currently the Microsoft Patterns & Practices Group has built the following Software Factories based on GAX/GAT:

- Smart Client Software Factory – guidance for Composite UI application block
- Web Service Software Factory – guidance for building services with ASMX or WCF
- Mobile Client Software Factory – guidance for building mobile composite smart clients
- Web Client Software Factory –guidance for building composite web applications

The designers included in the Visual Studio Team Edition for Software Architects are rather mature Software Factories. These types of domain specific languages can be built with the Domain Specific Languages Toolkit released last year (56). If you are interested in creating your own domain specific language, please refer to the detailed walkthrough on MSDN (57).

Microsoft's Future Plans

Microsoft continues heavy investments in domain specific languages. With the next release of Visual Studio (Codenamed ORCAS) it is planned to include GAX and GAT as a fixed part of the professional edition of Visual Studio and existing Software Factories based on GAX/GAT will be integrated into Visual Studio Team Edition for Software Architects as a graphical domain specific language including a contract-first message designer for web services.

Long-term plans (beyond ORCAS-timeframe) are the creation of DSLs for business capability modeling, requirements specifications and mapping to services. Figure 18 shows an example of how a DSL for specifying requirements could look like in a future release of Visual Studio.

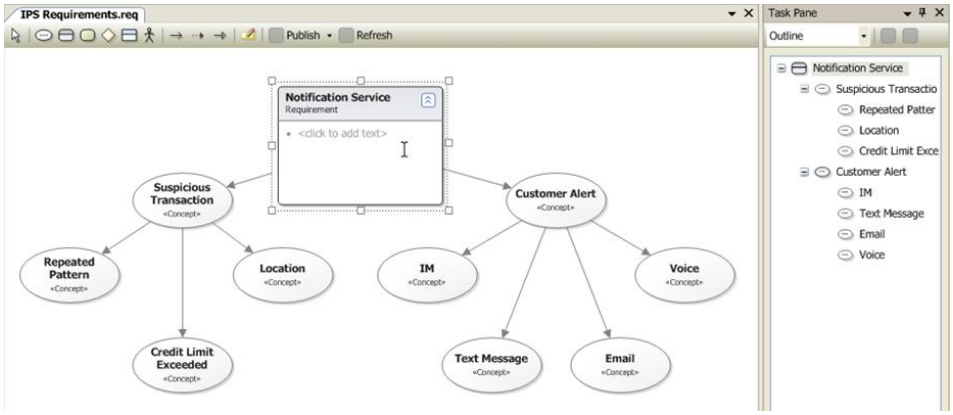

Figure 18: A sample DSL for specifying requirements

Table of Figures

Bibliography

1. Spark. [Online] 2006. http://sparklasvegas.wordpress.com/.

2. **O'Reilly, Tim.** Tim O'Reilly - What is Web 2.0. [Online] 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

3. **MIX, Microsoft.** MIX. *MIX.* [Online] 2007. http://mix06.com.

4. **TreeOfLife.** Tree of Life. [Online] 2006. http://www.flickr.com/photos/microsoft_spark/115177093/.

5. **Journal.** Journal. [Online] http://www.architecturejournal.net.

6. **Schwegler, Sehmi.** MSDN, Service Oriented Modeling, Part 1. *MSDN.* [Online] April 2006. http://msdn2.microsoft.com/en-us/library/bb245662.aspx.

7. **Homann, Ulrich.** Business Oriented Foundation of Service Orientation. *MSDN.* [Online] February 2006. http://msdn2.microsoft.com/en-us/library/aa479368.aspx.

8. **Homann, Tobey.** From Capabilities to Services. [Online] April 2006. http://msdn2.microsoft.com/en-us/library/aa479075.aspx.

9. **Microsoft Motion.** Motion Lite. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/aa479343.aspx.

10. **Microsoft.** Principles of Service Oriented Design. [Online] 2006. http://msdn2.microsoft.com/en-us/library/aa303159.aspx.

11. **Microsoft, WCF Versioning.** Data Contract Versioning. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/ms731138.aspx.

12. **Erlacher, Leitenmüller.** .NET Strategy Briefing. 2006.

13. **Wikipedia, Composite Application.** Wikipedia, Composite Applications. *Wikipedia.* [Online] http://en.wikipedia.org/wiki/Composite_application.

14. **Microsoft, WCF.** Windows Communication Foundation. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/netframework/aa663324.aspx.

15. **WS-*.** Understanding Web Services. *MSDN.* [Online] http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx.

16. **Microsoft, WCF Architecture.** WCF Architecture Overview for Developers. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/aa480210.aspx.

17. **Microsoft, WCF Contract.** How-to create a contract with WCF. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-gb/library/ms731835.aspx.

18. **Schwegler, Sehmi.** Service Oriented Modeling (Part II). *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-au/library/bb245673.aspx.

19. **Microsoft, WPF.** Windows Presentation Foundation. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/netframework/aa663300.aspx.

20. **Microsoft, WF.** Windows Workflow Foundation. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/netframework/aa663308.aspx.

21. **BizTalk.** BizTalk Server Developer Page. *MSDN.* [Online] http://msdn2.microsoft.com/en-us/biztalk/default.aspx.

22. **Microsoft, OBA.** Office Business Applications. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/aa479072.aspx.

23. **wikipedia.** BPM. [Online] http://en.wikipedia.org/wiki/Business_process_management.

24. **Greenfield.** Software Factories. [Online] 2004. http://msdn2.microsoft.com/en-us/library/ms954811.aspx.

25. **David Green, WF Platform.** Workflow Platform. *Architecture Journal.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/bb245670.aspx.

26. **PodCast Dave Green.** PodCast Dave Green. [Online] 2006. http://blogs.msdn.com/talk/archive/2006/09/29/interview-dave-green-17-microsoft-architects-forum-vienna.aspx.

27. **WF, Custom Activities.** Custom Activities, Tutorial. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/ms734563.aspx.

28. **ONE Case Study.** ONE Order Process Automation. *Microsoft Austria, References.* [Online] 2006. http://www.microsoft.at/referenzen/referenz.asp?id=1185.

29. **Wikipedia on UX.** Wikipedia, User Experience. *Wikipedia.* [Online] http://en.wikipedia.org/wiki/User_experience.

30. **WSS/MOSS.** Windows SharePoint. *MSDN.* [Online] http://msdn.microsoft.com/sharepoint.

31. **WSS Content Types.** Introduction to SharePoint Content Types. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/ms472236.aspx.

32. **ECMA Office Open XML.** ECMA Office Open XML File Formats. *ECMA International.* [Online] 12 2006. http://www.ecma-international.org/publications/standards/Ecma-376.htm.

33. **Wikipedia ESB.** Wikipedia Definition of ESB. *Wikipedia.* [Online] 2006. http://en.wikipedia.org/wiki/Enterprise_Service_Bus.

34. **Microsoft on ESB.** Microsoft on ESB. *MSDN.* [Online] 2005. http://msdn2.microsoft.com/en-gb/library/aa475433.aspx.

35. **Microsoft, ESB Guidance.** Microsoft EBS Guidance. *Microsoft.* [Online] December 2006. http://www.microsoft.com/biztalk/solutions/soa/esb.mspx.

36. **Carraro, SaaS Enterprise.** SaaS Enterprise Perspective. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/architecture/aa905332.aspx.

37. **Microsoft, Composite & OBA.** Composite Applications with the Microsoft Platform. *MSDN.* [Online] December 2006. http://msdn2.microsoft.com/en-us/architecture/bb220800.aspx.

38. **David Hill.** Composite Smart Clients. *David Hill.* [Online] 2005. http://blogs.msdn.com/dphill/articles/371327.aspx.

39. **Microsoft, CAB.** Composite UI Application Block. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/aa480450.aspx.

40. **Microsoft, SCSF.** Smart Client Software Factory. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/aa480482.aspx.

41. **AVL, TFMS Case Study.** Test Management Factory System Case Study. *Microsoft Austria Case Studies.* [Online] 2006. http://www.microsoft.at/referenzen/referenz.asp?id=1119.

42. **Szpuszta, CAB.** Designing Composite Smart Clients with CAB/SCSF. *Microsoft.* [Online] December 2006. http://www.microsoft.com/downloads/details.aspx?FamilyID=5f9a8435-1651-4be2-956d-0446a89a7358&DisplayLang=en.

43. **Microsoft, XAML.** eXtensible Application Markup Language (XAML). *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/ms747122.aspx.

44. **WF.** Windows Workflow Foundation. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/netframework/aa663308.aspx.

45. **Microsoft Expression.** Expression Tool Suite. *Microsoft.* [Online] 2007. http://www.microsoft.com/expression.

46. **Microsoft, MOF.** Microsoft Operations Framework. [Online] http://www.microsoft.com/technet/solutionaccelerators/cits/mo/mof/default.mspx.

47. **Microsoft & Amberpoint.** Microsoft and Amberpoint partnership. *2003.* [Online] http://www.microsoft.com/presspass/press/2003/feb03/02-11AmberPointPR.mspx.

48. **Microsoft PnP.** Microsoft Patterns & Practices Group. [Online] http://msdn.microsoft.com/practices.

49. **Microsoft, DSI.** Dynamic Systems Initiative. [Online] http://www.microsoft.com/windowsserversystem/dsi/default.mspx.

50. **Microsoft, SML.** Service Modeling Language, Specification. [Online] http://www.microsoft.com/windowsserversystem/dsi/serviceml.mspx.

51. **Microsoft System Center.** System Center Home. [Online] http://www.microsoft.com/systemcenter/default.mspx.

52. **Jack Greenfield, MSDN.** Software Factories on MSDN. *MSDN.* [Online] January 2004. http://msdn2.microsoft.com/en-us/library/ms954811.aspx.

53. **Wikipedia, DSL.** Domain Specific Languages, Definition. *Wikipedia.* [Online] http://en.wikipedia.org/wiki/Domain_Specific_Language.

54. **Software Factories.** [Online] 2004. http://www.amazon.com/Software-Factories-Assembling-Applications-Frameworks/dp/0471202843/sr=8-1/qid=1169592511/ref=pd_bbs_sr_1/105-3065150-5333224?ie=UTF8&s=books.

55. **Microsoft GAX/GAT.** Guidance Automation Extensions / Guidance Automation Toolkit. *Microsoft.* [Online] http://msdn2.microsoft.com/en-us/teamsystem/aa718948.aspx.

56. **Microsoft DSL Tools.** Domain Specific Language Tools. [Online] http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx.

57. **Microsoft, DSL Tutorial.** Creating Domain Specific Languages. *MSDN.* [Online] http://msdn2.microsoft.com/en-us/library/bb126278(VS.80).aspx.

58. **O'Reilly, Web 2.0, Compact.** Tim O'Reilly on a compact definition of Web 2.0. [Online] http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html.

59. **Platt, Compact Definition.** Michael Platt's compact definition of Web 2.0. [Online] http://blogs.technet.com/michael_platt/archive/2006/03/16/422247.aspx.

60. **Vista.** Windows Vista Developer Home. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/windowsvista/default.aspx.

61. **Michael Platt.** Michael Platt's Web Blog. [Online] http://blogs.technet.com/michael_platt.

62. **Michael Platt (new).** Michael Platt's new Web Blog. [Online] http://michaelplatt.net/blogs/architecture/default.aspx.

63. **Platt, Enterprise 2.0.** Michael Platt's Web Blog, Enterprise 2.0. *Michael Platt.* [Online] https://blogs.technet.com/michael_platt/archive/2006/06/06/433461.aspx.

64. **Microsoft, WCF.** Data Contract Versioning. *MSDN.* [Online] 2006. http://msdn2.microsoft.com/en-us/library/ms731138.aspx.