Achmea streamlines application development and integration

Danny Greefhorst and Patrick Gehner

Achmea is in the process of standardizing application development, integration and the accompanying infrastructure in order to lower costs and increase the agility of the business. Architecture is needed to guide Achmea in this journey, in order to prevent developing tomorrow's legacy. A Software Reference Architecture has been developed for this purpose, which aims to guide the development of new applications and their integration with other applications. This article elaborates on this Achmea Software Reference Architecture: the background, concepts, content and process.

Business context

Achmea is the name of a conglomerate of Dutch insurance companies. In the last twenty years Achmea grew through mergers and business integration and collected a wide range of IT infrastructures and applications in this process. Typically after a merger a new business unit was formed and the enterprise IT landscape was enriched with another IT stovepipe. The Achmea business units are characterised by a high degree of autonomy. Synergy was promoted between the business units but no attempt was made to integrate at an enterprise level. This loosely coupled enterprise could successfully absorb many mergers in a short period of time. After settling in the new context the process of enterprise integration started and the first service units were formed. One of these service units was Achmea Active, the internal IT service provider, for developing business applications and delivering the technology infrastructure. The Central Technology Office (CTO) of Achmea Active was responsible for IT purchase and IT architecture at the enterprise level, an important step toward mature IT governance. The recent merger with Interpolis will place CTO in the new Group IT Services.

Achmea

What started in 1811 as a cooperative insurance society 'Achlum', founded by a few local farmers in a small town in the north of the Netherlands, became a leading financial service provider in the Netherlands. A history of mergers lead to a loosely coupled fleet of companies working together and expanding into Europe. Achmea plays an important role in the Eureko Group, a privately-owned financial services group, whose core business is insurance, and which has operations in ten European countries. Achmea offers businesses, institutions and consumers a broad range of insurance, banking and mortgage products. Achmea links other services to these products to enhance their value and to provide greater convenience for the consumer. These services include assistance at home and abroad, health and safety services, absenteeism prevention and workplace reintegration services. Achmea also administers pension schemes. Achmea is known in the market for services which 'unburden' its customers. The most important brands are Centraal Beheer Achmea, Zilveren Kruis Achmea, Avéro Achmea, FBTO and, after the last merger, Interpolis. In 2005, Eureko registered a before-tax result of EUR 826 million. This success follows the strong result of EUR 486 million that was achieved in 2004. Achmea contributed EUR 489 million to the net income of the Group. A total of 19.000 employees are working for Achmea in more then ten business units and service units.

Business challenges

The business of Achmea is all about providing services assembled from services of the different business units. A `service oriented´ business like Achmea could benefit from a 'service oriented' IT. Service based architecture could help in attaining strategic goals such as:

- **Improving commercial vitality** – IT as the business enabler
- **Accelerating business service development** – Time-to-Market reduction
- **End-to-end services delivery** – More control of the supply chain is key to survival in the market
- **IT cost reduction** – Achmea now supports nearly all the IT solutions in the market
- **IT standardisation** – IT convergence is needed
- **Reuse business logic** – Sharing information systems is now too difficult

The IT strategy and policy of Achmea is directed towards reducing the diversity in order to simplify the management of the IT infrastructure. Many software development environments are used to develop and maintain the Achmea applications for even more separate runtime environments to deploy the applications. The complexity is huge when application integration is needed to support business units working together in delivering new end-to-end services. There is too much complexity on every infrastructure layer. The IT challenge consists of reduction of complexity through consolidation and optimisation.

Active CTO recognizes the convergence in the IT market towards service orientation and the opportunity to streamline application development and integration for Achmea. CTO decided to prepare the paradigm shift needed to reach the goal of a standards based, sustainable and adaptive IT support for the business.

A new focal point

When Achmea Active became responsible for the technical infrastructure it started a bottom-up approach in the reduction of the IT complexity. The wide area network that connects the isolated infrastructure islands was the first part of the new enterprise infrastructure and an outsourced solution was selected. The consolidation of the many different workplace infrastructures was the next big step, introducing one uniform Achmea workplace. The new Achmea workplace services consist of a desktop or laptop with all the basic office, communication and groupware services, access to applications from Achmea office locations and supporting employees working at home. Finally the fourteen data centers were moved into a twin-data center after consolidating servers, storage and infrastructure management tooling. All this was done by carefully avoiding any big impact on the applications and thereby the business.

Further optimisation of the technical infrastructure required consolidation on the application infrastructure level. The application infrastructure consists of the different application containers to develop and run the Achmea applications. This area is dominated by IT professionals with strong feelings about the future of their IT product. Without a new common focal point Achmea was facing endless discussions and time consuming product evaluations selecting the best product, when it already had all the good products.

Service orientation and the emerging standardization provided a way-out. A new focal point combined with some IT rethinking could make the difference. Key elements of the approach:

- **'product suite' thinking instead of 'best-of-breed' thinking** – Outsourcing the integration of the core products to the vendor of the IT ecosystem. An ecosystem is defined as a set of products from one vendor that are designed to be used together.
- **Service orientation** – Leveraging the movement in the IT market; adopting standards, proven methods and best practices. End the search for the perfect tools and take advantage of the service oriented evolution of the present tools.
- **Mature IT architecture** – Development of an enterprise IT architecture and implementing enterprise IT governance where key ingredients for making the paradigm shift to service orientation. The 'integrated architecture framework' (IAF) from Capgemini was adopted to define the **Achmea Architecture Approach.**

This provided Achmea with the roadmap to service orientation and the next target: optimising the application infrastructure (also see figure below). This meant that a new service oriented infrastructure with connections to all the legacy infrastructure islands needed to be built. This application infrastructure should facilitate the optimisation of the business applications: the information infrastructure of the organisation.

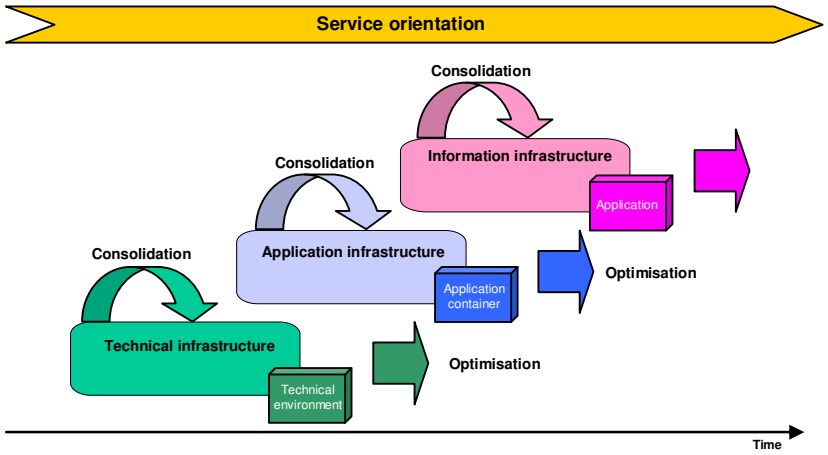

Figure 1 Roadmap to service orientation

Preparing for the road

Merely defining a service oriented roadmap is not enough to start the journey. Achmea decided to form a small group of specialists with knowledge of Enterprise Application Integration and Development and the two major IT ecosystems (IBM and Microsoft). This core-team of service oriented pioneers found an approach for introducing service orientation on an enterprise level. They composed relevant standards and available best practices of the vendors and adapted them to the Achmea context. The resulting Achmea 'Software Reference Architecture' (SRA) was the means to accelerate service orientation. A team effort and team members that became change agents did the trick. The team started early on in the year 2005 and, with the support of specialists from IBM and Microsoft, delivered the SRA in two iterations. Both iterations used the same work method:

- **Centered around SRA core-team** – Accelerated the preparation and acts as a group of change agents.
- **Workshop-based** – In a prepare-discuss-amend-conclude cycle , the reference material was moulded to fit the Achmea context.
- **One ecosystem at a time** – Focus was essential to deliver the results. Exploring service orientation in the context of one ecosystem (IBM, Microsoft) at the time and addressing only the subject important for the next evolution step.
- **SRA 1.0** – Scope: on-line J2EE applications. Focus on IBM ecosystem and service orientation in the context of online J2EE applications. Workshops on overall architecture, application integration, user interaction, process choreography, security and logging. Bootstrapped with IBM best practices (Redbooks and reference architectures)
- **SRA 2.0** – Scope: service-oriented applications. Focus on integration of SRA 1.0 with the Microsoft ecosystem. Workshops to broaden the scope of service orientation including .NET applications, information management (BI/datawarehousing) and batch processes (batch/files). Enriching the architecture with the Microsoft best practices (Microsoft .NET application architecture)

The architecture

The architecture was developed in line with the overall Achmea architecture approach. The accompanying architecture framework distinguishes business, information, information system and technology infrastructure viewpoints. The scope of the architecture is the information system viewpoint. That is, it provides guidance in the development, integration and acquisition

of information systems. The architecture approach also distinguishes conceptual, logical and physical levels of abstraction. These levels were adopted as the main structure of the architecture document. The content itself consists of concept, principles and guidelines. Concepts form the conceptual architecture and are an important part of the architecture since they define a common language. They basically prevent endless discussions about terms and their meaning, which is a good timesaver in itself. The principles state the overall direction for all architecture and design decisions that are made for applications within the scope of the architecture. They form the umbrella for the guidelines. Guidelines are explicit directional statements that can only be deviated from when there is a good reason. They should however not be interpreted as rules that should never be broken; they are meant to stimulate explicit reasoning and discussion. This also differentiates the reference architecture from solution architectures; solution architectures allow less freedom and can really be seen as high-level designs. The rest of this section steps through the various parts of the architecture: the principles, conceptual architecture, logical architecture and physical architecture.

Principles

The principles are meant to provide overall direction. They were chosen because they define new ways of working that will improve application development and integration activities. Some were defined upfront; others were found along the way, while defining the guidelines. The principles defined by the architecture are:

- **Applications operate full-time in real-time** – This principle bubbled up when looking at the future role of batch applications. It recognizes that Achmea needs to support channels such as Internet that do not follow opening hours. Also, competition requires Achmea to process requests straight through, as much as possible.
- **Applications conform to a Service Oriented Architecture** – This is really the most important foundation of the architecture. A service oriented style of development and integration is no longer just an option; it is mandatory for large and complex organizations to keep up with competition and keep application management costs at an acceptable level.
- **Applications are structured into manageable components** – Service oriented architecture does not make component based development superfluous. On the contrary; while service orientation focuses on managing application integration, component orientation focuses on managing the internal structure of applications. Following from this principle, the architecture provides a lot of guidance in the types of components that exist, and the responsibilities that they should have.
- **Applications respect logical units of work** – This principle was added in the second version of the architecture when it became clear that a number of guidelines really revolved around the concept of a logical unit of work (transaction). Particularly, the selection of an appropriate service interface and communication style very much depends on these units.
- **Applications conform to standards** – Conforming to open, or else de facto, standards really increases the maintainability of applications; it eases integration with tools and other applications and prevents a lock-in to proprietary solutions. This is why it was explicitly added as a principle to the architecture.

Conceptual architecture

Since the architecture is service-oriented and component-based, a lot of terminology revolves around these two architectural styles. With respect to services, the architecture needs to conform to the terminology of the Achmea architecture approach that deals with business services, information services, information systems services and technology infrastructure services. The good thing is that the architecture approach seems to be designed with service oriented architecture in mind. The bad thing is that the terminology does not entirely fit onto a software interpretation of the term "service"; a piece of functionality that can be invoked programmatically by sending a message to it. Information system services are defined broader than this; they also include functionality that is provided to the user through a user interface (see the figure below). The SRA definition of service also includes technology infrastructure

services, especially since it is hard to make a clear distinction between application-specific functionality and infrastructural functionality. Another problem that arose in the definition of the term "service" was its use in Web Services technology (e.g. WSDL). In WSDL a service is a collection of interfaces that provide operations. Since the SRA definition is much more in line with an architectural view of information systems, it was decided to make the distinction explicit by describing the relationship. A service as defined in the architecture is translated to one or more operations in a service interface of a Web Service.

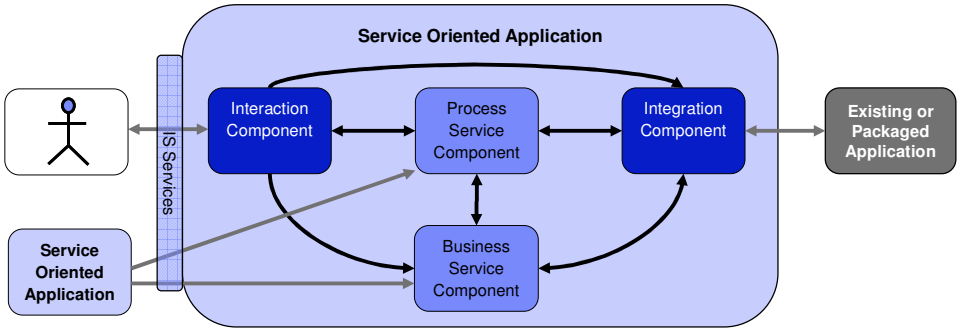

Figure 2 Concepts

Another large part of the conceptual architecture is the definition of the various types of components that can be discerned. Interaction components manage the interaction with the end-user. Process service components manage (cross-business domain) business processes. Business service components manage business data (e.g. the "customer" business service component). Integration components manage the communication with existing- and packaged applications. In addition, ETL components are distinguished in the information management domain. ETL components coordinate the process of extracting, transforming and loading data from one data store to the other.

You may wonder about the explicit use of the word "service" in the types of components. The word was added in the second version of the architecture when discussions arose about the granularity of components and their relationship to Web Services. It was recognized that service-oriented components should be coarse grained and are typically implemented by a Web Service. This is where the term "service component" came from; it is a coarse grained component that provides services and that includes smaller components. Actually, the architecture also defines technology infrastructure service components that provide a generic service interface on top of technology infrastructure. The smaller components within the service components have similar names (process components, business components and technology infrastructure components), but do not provide services and can exist together within a service component. For example, the "customer" business service component may include a "person" business component and a "customer service" process component.

Logical architecture

The logical architecture consists of a large number of guidelines that are completely implementation independent. A part of this logical architecture is a component model that describes the layers and responsibilities of the various types of components. It was constructed by combining the Microsoft Application Architecture for .NET with generic and J2EE architecture and design patterns. The figure below provides an overview of the logical architecture, including the component model. For example, a business component consists of a façade, one or more business objects, zero or more data carrier objects, zero or more data access objects and zero or more service agents.

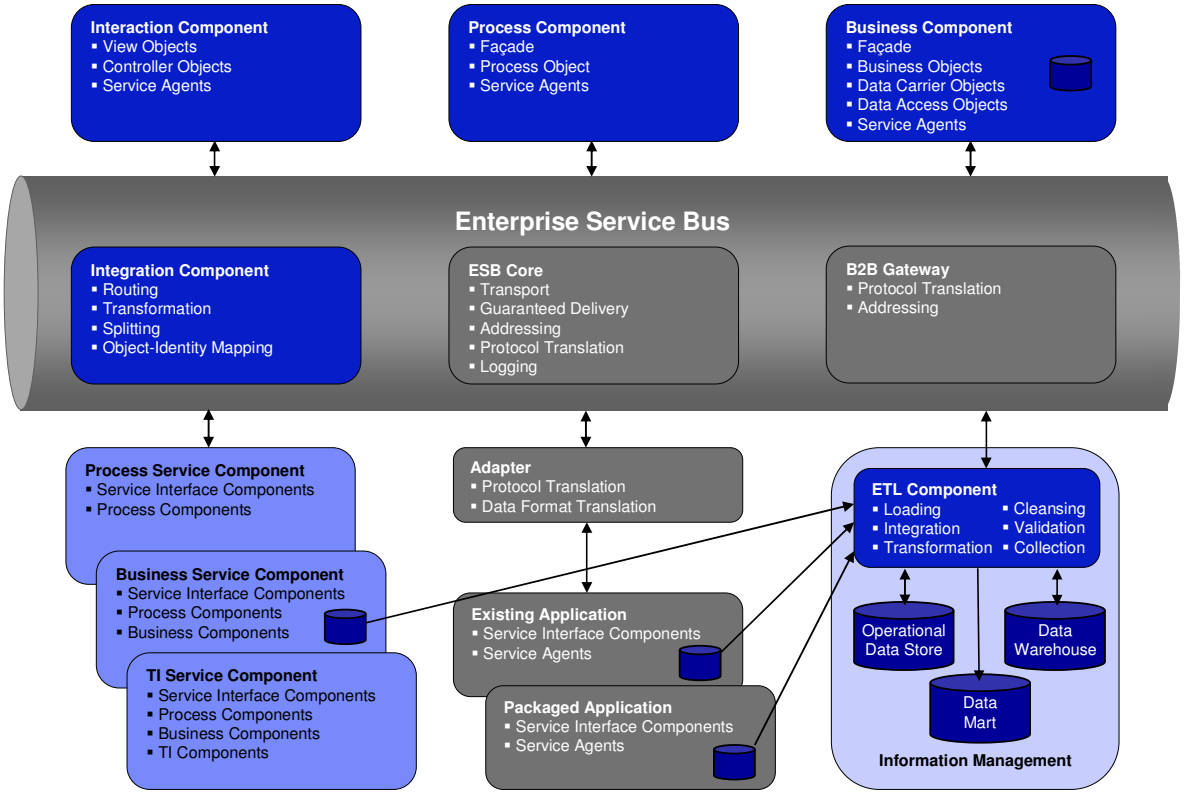

Figure 3 Logical architecture overview

To provide some high-level insight into the logical architecture, a number of high-level statements are presented that actually form the section headings in the architecture document. They provide a logical grouping for the guidelines.

- **Services should be defined in a reusable and stable manner** - Services really form the foundation of a service oriented architecture; their reusability and stability actually determines the success of the architecture as a whole. That is why a lot of attention needs to go into the definition of service interfaces. As a general guideline, services should be defined independent of a specific service requester.
- **Transactions should determine the style of communication** – As mentioned before, transaction boundaries very much influence the style of communication that is most appropriate. As an example, if the invocation of a service should be part of the overall transaction, a request-reply style of communication should be used. This is because a reply is needed to determine whether the invocation was successful, and transactions should only include short-lived operations.
- **Communication should rely on the Enterprise Service Bus** – The Enterprise Service Bus is nothing more than a term that includes all integration middleware that is needed. It includes a core that handles transport, including location transparency. It also includes integration components that can mediate between service-oriented applications and existing applications. Note that using the ESB is not equivalent to having an intermediary for every service invocation.
- **Files should be treated similarly to messages** – It may sound obvious, but a file is really very similar to a message and should be handled accordingly. This means that file exchanges should be designed service-oriented; a file format is really a service interface.
- **User interfaces should be chosen and designed for their users** – User interfaces are the primary place of contact for users and should be designed with care. They should align with the work of users, but should also be structured neatly internally so they can evolve with future user requirements.
- **Disparate user interfaces should be integrated in a portal** – Achmea, like most financial institutions, has a heterogeneous application landscape that needs to behave in an

integrated fashion to users. Integrating disparate applications at the user-interface level is best performed in the context of a portal, since that provides a number of generic facilities.

- **Business processes should be explicitly defined and implemented** – Business process logic is typically embedded in application code. From a reusability and maintainability perspective this logic should be separated, and defined explicitly in a process modelling tool. This is especially relevant in the context of a service oriented architecture.
- **Batch processes should transform into on-line processes** – Batch thinking stems from previous times where computers had all sorts of limitations. Nowadays, most batch processes can be performed on-line using the services that are already available in the organization.
- **Business logic and data should be managed carefully** – The core of the organization is the business data and the logic that manages this data. Inconsistencies can frustrate organizations and should be prevented. Replication of data and functionality should only be allowed under special conditions.
- **Business Intelligence should work upon specific data organizations** – Business Intelligence processes are meant to turn business data into information. They often require aggregated and integrated views of business data, and should therefore depend on data stores that are specifically designed for that purpose.
- **Sensitive data should be protected** – Security is an issue, especially in a service-oriented architecture. Security measures should be taken both inside and on the outside of the organization because organization borders disappear and most security attacks come from the inside.
- **Systems management events should be exposed** – Applications should not only be designed for business end-users; there are also end-users in the IT department that need to manage the application. Explicit attention should be paid to identifying and exposing those events that are relevant to these actors.

Physical architecture

The physical architecture focuses on application integration, and on Web Services standards in particular. The prime motivation is that Web Services standards are open standards that have gained industry acceptance. In other words; vendors position Web Services as the preferred option for integrating heterogeneous systems, and have optimized their software products for it. Standardizing services on this technology makes them available throughout the organization, since they are available in a large number of technology environments.

From an architecture perspective, the Enterprise Service Bus can be seen as consisting of three separate busses; a SOAP/HTTP bus, a SOAP/MQ bus and an FTP bus. Since Web Services interoperability does not come automatically the Web Services Interoperability (WS-I) consortium has defined standard profiles. The WS-I Basic Profile sets a baseline for a number of Web Services standards, and provides guidance on how to use them. Conformance to WS-I Basic Profile is thus mandatory for all SOAP-based communication. Note that SOAP/MQ is not a formal WS-I Basic Profile standard. WebSphere MQ is however needed since HTTP does not support queuing; services that require queuing are thus exposed through WebSphere MQ. Also, WS-I Basic Profile should be adhered to as much as possible. The FTP bus is where files are exchanged, which may still be needed for bulk transfers or because it is mandated by existing, packaged or external applications.

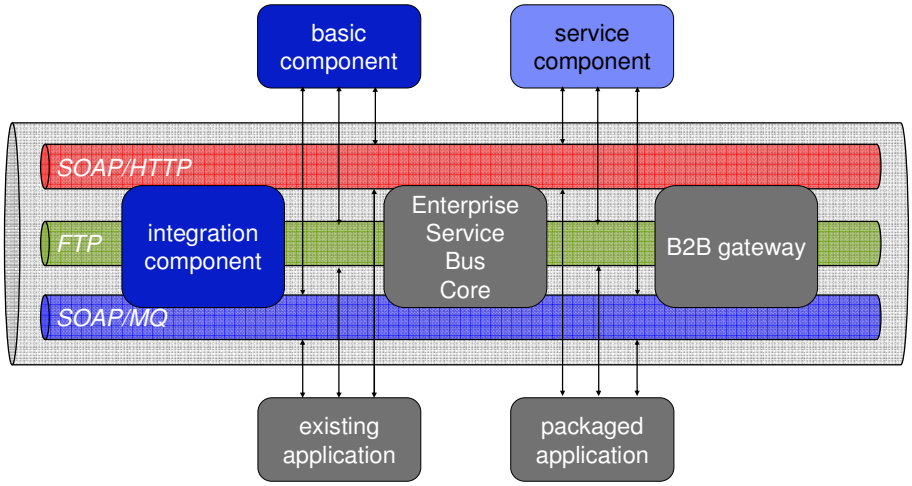

Figure 4 Physical architecture

Getting it on the road

Good preparation is only the first step; implementation of all these refreshed and new guidelines requires more than a comprehensive document. A 'big bang' scenario is not an option in the context of Achmea; rewriting millions lines of code is simply not possible. The new way of working is building composite applications, reusing legacy code and integrating these with newly bought or developed code, guided by SRA guidelines. This also requires a transformation of the software development approach. This transformation involves training and convincing developers, application architects and other stakeholders. It also requires implementing a suitable service oriented application infrastructure. Achmea started a number of important steps that are a prerequisite for building applications based on SRA and for modernizing the application portfolio:

- **Develop the base service oriented application infrastructure** – Transformation of the present message based infrastructure to implement a fully fledged enterprise service bus. Implement the first increments of an enterprise application portal.
- **Define J2EE and .NET reference architecture** – Translation of the SRA guidelines to the specific development toolsets and construction guidelines to simplify the SRA interpretation for the developer.
- **Train the project architects** – The reference architecture is used by the project architects to create the solution architectures for their projects. By training this group, the number of service oriented change agents will grow.
- **Implement a reference application** – An environment to experience the essence of a composite application and the way to specify, build, deploy, use and maintain it. In lab exercises different groups can see and feel the SRA guidelines without the normal complexity of a business application. Showing the elementary application patterns for Achmea aids implementation conforming with the SRA.

At this moment in time, project architects have applied the principles and guidelines in their project architectures for about one and a half years now. Infrastructural components such as a new message broker and portal infrastructure have been put in place and projects are underway for migrating to this new infrastructure. The architecture has been successfully used as an architecture review framework, providing direction in the improvement of existing applications. Due to the recent merger with Interpolis the organisation will be transformed, and new architectures will be defined. At the highest levels, Business Information Plans are developed that provide an end-to-end (from business to technical infrastructure) view of the organisation. These Business Information Plans will incorporate elements from the SRA. At a lower level the SRA will be merged with the software architecture of Interpolis.

Lessons learned

The authors learned a number of lessons in the process of defining and implementing the architecture. Most prominently, they learned that service orientation is a paradigm shift; even the most experienced IT professional has to work hard to make the shift. The difficult part is that it's not something completely new: it is basically a lot of old rules that are placed in a new context and with an additional focus on open standards. For example: the current approach of application integration may look service oriented at first glance, but is in fact the opposite. It requires central integration logic for every new connection between service providers and service consumers.

Discussion arose with respect to the extent in which the architecture guidelines are mandatory. In the view of the authors guidelines are best practices that should be followed in 90% of the situations. Room should be left to deviation when appropriately motivated. In the context of service orientation, compliance of applications and products to standards should however not be taken lightly; these standards are fundamental in facilitating integration.

There is still no 'one-size-fits-all' architecture. Based on the standards and best practices a company must make its own architectural choices on an enterprise level. These choices are not necessarily the best in all respects, but standardizing and guarding them has value in itself as it leads to uniformity and adaptability. Architecture and mature IT governance are fundamental requirements in order to attain these qualities in practice.

The design of the business in terms of services helps in aligning business and IT since IT support will be delivered in IT services. IT can start on a transformation towards service orientation by making sure the necessary infrastructure is in place and people are trained. At some moment in time however, business must buy in to the vision otherwise the true benefits of service orientation are not attained.

It has proven to work well when you start the preparation with a small internal group of experts and use the support of your main software vendors, adopting the standards and the available best practices. The central IT group is in the lead helping the development and deployment organisations to make the necessary changes.

A software reference architecture alone is not enough. Further detailed architectures and guidelines (e.g. J2EE and .NET specific) are needed to simplify the use of the guidelines in practice. Also, a lot of other activities are needed to transform the organisation into a service oriented organisation. This starts at the business, that needs to buy into the approach, and ends with application and infrastructure developers that need to adapt their ways of working,

Conclusion

Service orientation provides a focal point for optimization of the technical, application and information infrastructure. The paradigm shift to benefit from the emerging business value of service orientation is however not easy to achieve. Architecture and IT governance must be mature to really benefit from the transformation to service oriented development of applications.

Service orientation is an instrument to provide clear directions on the journey to an optimized IT infrastructure and certainly not the Holy Grail of IT; you can still build your tomorrow's legacy. However a service oriented application infrastructure is a necessary pre-requisite for software reuse, composite applications and creating solutions for the adaptive enterprise.

Service orientation provides a focal point and facilitates convergence. It's not a goal in itself, and it is certainly not something to start for unprepared.

Danny Greefhorst
IBM (now Yellowtail)
dgreefhorst@yellowtail.nl

Patrick Gehner
Achmea
Patrick.gehner@achmea.nl