

Service-Oriented Architecture Applied

by Bartek Kiepuszewski, Michal Paluskiewicz, and Borys Stokalski

INTRODUCTION

Another three-letter magical word makes its way to marketing presentations, industry media, conference rooms, training curricula, and — of course — to IT budgets. Should you bother? As with every such concept, the answer is both yes and no. There is definitely nothing magical about service-oriented architecture (SOA). The basic engineering concepts behind SOA are well known and understood since — at least — the announcement of Distributed Computing Environment (DCE) by the Open Software Foundation. On the other hand, if you find these engineering concepts concerning the management of highly distributed software environments appealing but too esoteric to implement in the “real world,” then you should be aware that, as Bob Dylan used to sing, “the times they are a-changin’.”

There are important reasons to consider SOA a feasible architecture option for any mainstream IT organization. Among the reasons are important global trends — such as the growth of the Internet and proliferation of electronic channels; the advent of widely accessible, cross-technology application integration standards, such as Web Services; the architectural

evolutions of the large packaged software vendors; and — in many cases — the investment many organizations have made in enterprise application integration.

But will SOA work for you and your organization? The decision to apply any enterprise architecture pattern requires careful consideration and is always context specific. We hope that by presenting case studies of successful (although not ideal) real-life applications of SOA, along with some generalization ideas, we will help readers in taking their own decisions.

Onet.pl — THE E-COMMERCE NETWORK

After the collapse of the Internet bubble in 2001, even the most successful dot-com initiatives in Poland were suspected of being potential business disasters.

Onet.pl — the most popular horizontal portal on the Polish market — was looking for ways to strengthen its revenue streams in order to retain the confidence and support of its stakeholders.

An important line of revenue in those days of declining e-marketing opportunities, growing disappointment with the barter deals typical of the Internet economy, and falling

valuation of Internet business was e-commerce. Onet.pl decided to focus its e-commerce strategy on developing a strong alliance with a growing community of Internet retailers and to serve them as the main supplier of e-customer traffic. For every business hoping to get customers from the Web, customer traffic is a critical asset, compared only to financing. Here Onet.pl could help as an undisputed leader of the Polish Internet.

Onet.pl obviously offered its visitors an e-mail, but the then-existing functionality and architecture were no longer up to the requirements. Onet.pl wanted an e-commerce platform that would enable an integrated product catalog, tracking of e-customers’ purchasing behavior, and options for future unified payment and checkout procedures. On the other hand, Onet.pl also wanted e-shops to remain independent in terms of their approach to customer experience, Web design, and technology, as well as in terms of liability for transactions.

The requirement of technological independence was an important business assumption. Onet.pl understood that the process of integrating e-shops has to be simple, fast, repeatable, and transparent for customers. As we will see later,

a service-oriented architecture allowed Onet.pl to achieve these objectives reasonably well for most “mainstream” e-shops, although some of them required either simplified handling or customization of the generic integration approach. The project involved also an important skills transfer component, as Onet.pl wanted to standardize on Web application server technology and implement an agile software process. These were considered important capabilities for future growth. The mix of requirements and strategic directions resulted in the idea of an e-commerce value network, built on top of a software platform architected according to an SOA pattern.

When our company, Infovide, embarked on a project aimed at creating the e-commerce value network, it turned out that Onet.pl was in the middle of a technology transition from the popular PHP-style way of creating Web portals to increasingly mature Java technology and Java-based application servers. Apart from that, we found a couple of legacy systems, such as software responsible for serving banners and ads (the AdServer) written primarily in C/C++. All that provided us with a great stimulus not only to expose services to external partners, which seemed a very natural thing to do in a B2B system, but to base the whole platform around the concept of services. This way we could deliver a set of internal services to a group of Onet.pl programmers that produced the PHP-based shopping mall front end as

well as integrate with all legacy systems.

“Always do the simplest thing that can possibly work” is one of the main principles of Extreme Programming. With all the advantages that an SOA-based architecture provides, it is not exactly the simplest way of doing things. Running the project under strict deadline, we deployed an XP-based methodology, and as a result, we had to find the fine line between an architecture that is the “simplest possible” and one that is so simple that it loses important advantages of SOA.

Taking into consideration all of the above, we have applied the following architectural principles for the project:

1. Services are the basic building blocks for both external and internal communication with the main modules of the platform. “Internal” GUI applications for system administrators are built using a simpler, more traditional, layered architecture with the Web client accessing EJB-based services through the direct, RMI-based interface.
2. There is no single technological standard; hence a Web services stack of protocols is deployed as the principal middleware for the architecture.
3. Service-level agreements (SLAs) for the services are carefully defined and understood so that the client of the service can take some preemptive actions when its SLA is more stringent than

With all the advantages that an SOA-based architecture provides, it is not exactly the simplest way of doing things.

what the service itself can provide. As an example, the PHP-based shopping mall front end had to implement some aggressive caching techniques to meet the requirements of a robust, 24/7 Internet portal.

The end result is the architecture depicted in Figure 1.

Key Lessons

- Using two distinct technologies for the GUI and service layer — apart from the technical challenges it gave us (after all, Web services technology was at its inception at the time of the project) — proved to be a great enforcer of the SOA architecture. No shortcuts were possible, and all the services had to be designed with great care for granularity and state management in particular.
- Integrating many different shops into the shopping mall gave us the whole spectrum of possible integration scenarios. Big shops exposed their interfaces for us, for which we had to write our own adapters. Medium-sized shops had to adjust to our interfaces. Web services technology proved to be a too big a hurdle for small shops to take on, and for these we had to resort to a simple

data file download from their HTTP servers.

- SLAs for both internal and external services turned out to be one of the most underappreciated issues at the beginning of the project. There can be no mature SOA if SLAs are not properly understood and managed.

Omnix — THE MOBILE SERVICES PLATFORM

During last three years, European mobile network operators (MNOs) have started working on product strategies based on nonvoice mobile services. The key drivers for these attempts are business threats (such as the expected decline of average revenue per user

and increasing customer churn) as well as opportunities (new revenue streams from innovative services delivered via mobile terminals).

This new model has already proved its viability on the Japanese market. NTT DoCoMo serves as a role model, being a spectacular business success achieved through the creation and exploitation of mobile services. MNOs view their existing customer base not as mere subscribers to mobile communication services, but as a highly interconnected network of relationships among individuals, communities, and organizations. Only a small part of these relationships is today explicitly mapped to billable services delivered via a mobile communication infrastructure.

One of the early movers into the new area of mobile business in Europe has been PTC Era — the leading MNO in Poland. In November 2002, PTC launched Omnix, the first mobile nonvoice service in Central and Eastern Europe, and one of the first in the whole of Europe. Omnix started by offering the users of mobile terminals (a broad range of mobile phones and PDAs) a portfolio of “infotainment products,” such as multimedia clips, news services, business directories, location-based services, mobile games, multimedia messaging service (MMS), e-mail, and instant messaging.

The compelling results of DoCoMo have been founded on a large network of companies that develop, market, and deliver services via the DoCoMo network. Over 40 million DoCoMo subscribers can choose mobile services from over 77,000 i-mode¹ sites. DoCoMo partners are often the ones who take the business risk associated with bringing

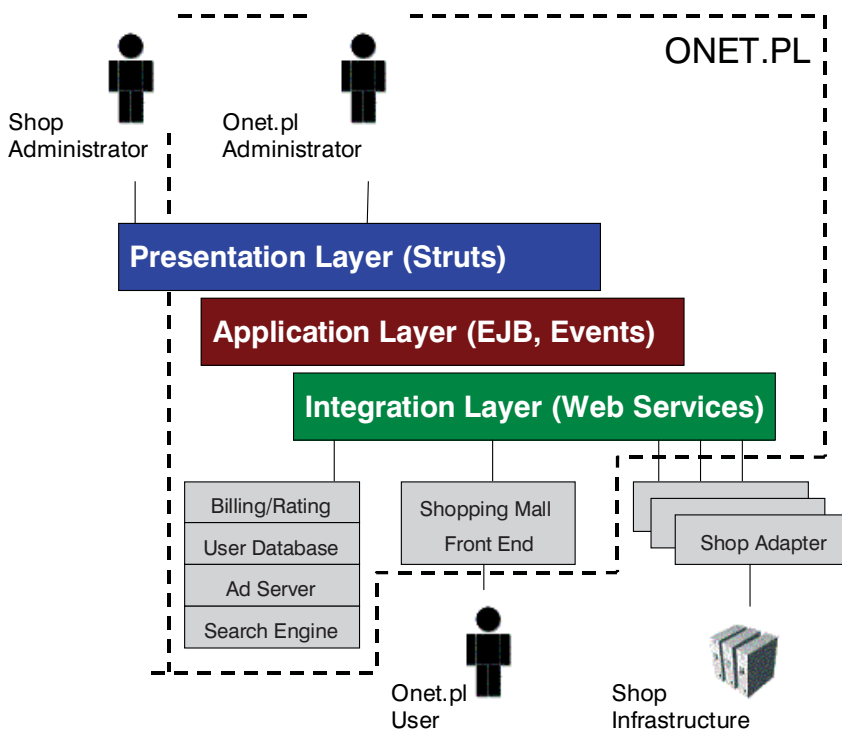


Figure 1 — Onet.pl architecture.

¹Technically i-mode is an overlay over mobile voice communication network operated by NTT DoCoMo, a mobile network operator founded by NTT (Japanese telecom). While the voice system is “circuit switched” (i.e., you need to dial up), i-mode is “packet switched.” This means that an i-mode user is “always on” within the range of the network. DoCoMo provides handset manufacturers with strict technical and usability specifications, allowing each DoCoMo subscriber to use the handset not just as a voice communication device, but also as a mobile Internet terminal, navigating multimedia content delivered by i-mode sites.

innovative services to market and making them profitable. This observation guided the strategic choices of the team responsible for developing nonvoice services at PTC. The team's basic assumption was that Omnix will aggregate the infotainment content from third-party suppliers. To achieve initial success, PTC needed to quickly assemble a portfolio of business partners capable of delivering quality services for Omnix. Sustaining the success required consistent growth of the partner network and a profitable service portfolio.

In order to encourage partners and provide them with the means to mitigate risks inherent in a business innovation based on new technology, PTC contracted with Infovide to architect and build the Infotainment Platform — a software environment enabling the creation, aggregation, and delivery of mobile services. The main goals of the platform are: quality delivery of services for customers, consistently low cost and short time to market for potential services, and the elimination of technical barriers for potential content suppliers.

The goals have been achieved by adopting a number of key architecture principles:

1. Services are the key building block of the platform. This approach involves both the internal platform services, the portfolio of services provided as basic building blocks for content suppliers, and services representing infotainment products implemented by content suppliers.

2. A single technological standard (J2EE) is used for the construction of platform services and services implementing infotainment products.
3. Platform services are layered to separate platform logic, infotainment product logic, user interaction, and integration with mobile network infrastructure (such as short message service [SMS] or terminal localization services).
4. Platform services should be generic and should not contain any logic specific to the content application (infotainment service).
5. Developing infotainment services should not require in-depth J2EE knowledge. Moreover, even poorly designed microapplications (as the infotainment service implementations have been called) should not be able to affect the execution of other microapplications or the platform itself.

To achieve initial success, PTC needed to quickly assemble a portfolio of business partners capable of delivering quality services for Omnix.

The resulting architecture is presented in Figure 2.

Selecting SOA as the fundamental architecture pattern for the Infotainment Platform was an obvious choice. The Omnix service has been on the Polish market for more than 18 months now, and we can safely say that the architecture serves its purpose very well.

Omnix has currently over 200 categories of services delivered by a variety of content partners, the time to market of new service implementation is yet unmatched by PTC competitors, and the

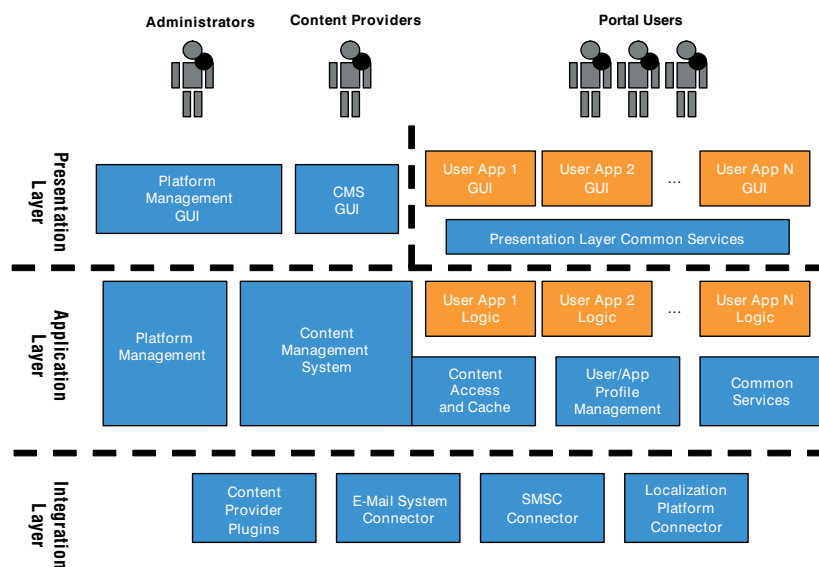


Figure 2 — Infotainment Platform Architecture

number of subscribers is growing at a double-digit rate. Thanks to the use of adaptive application development methodologies (based on XP), it took less than five months to architect, design, build, and deploy the Infotainment Platform with its initial portfolio of infotainment products.

Design of interfaces is the most important part of the whole project.

The innovative character of the project also delivered a number of lessons and observations concerning the things to be avoided in similar projects:

- In the first phase of the project, each XP cycle delivered a “vertical slice” of the system — a small set of business functionality — from user interface to database interactions. This narrow, functional focus created some architecture inefficiencies. The functions of the platform, as it turned out later, have some degree of interdependence, which influenced the overall architecture. We later changed the process, starting with analysis of functional interdependencies and using this output to plan development cycles so that the “core” functions are designed and developed in the first place.
- Short deadlines caused fast and parallel development of vertical parts of the system. Sometimes developers took shortcuts, and some parts of the system needed refactoring after the end of the project. We also noticed that if one developer is responsible for creating both a service and one of the service’s clients, he tends to mix the responsibilities of these two parts.
- It is often not easy to find the proper balance between generality of services and their ease of use. Generic services are easier to manage and look better on diagrams, but they require more work to customize them for use in a specific client context.
- The service-oriented principle of separating interfaces from their implementations means that version control is more complex than in traditional approaches. In particular you must manage versions of interfaces separately from versions of implementations (while at the same time keeping track of which implementations realize which interfaces). Design of interfaces is the most important part of the whole project and should be performed with thorough consideration of internal dependencies. In the best-case scenario, once an interface is designed, it should never be changed.
- The system was created so fast that many content providers could not keep up with porting their existing content delivery mechanisms to Web services.

As a result, we had to develop plug-ins that enabled us to get content from some of the providers in a more traditional way. The plug-ins were clients of standard B2B services, so there was no need to change the SLA monitoring or other parts of the system. But if you are adopting new technologies within a partner network, you should be prepared for the fact that some of the partners will miss their deadlines, and you may want to prepare workarounds as part of the project.

SOA ROADMAP

SOA definitely has its strengths when used to solve the right problems. It can also be misused when the purpose and means change places. SOA is by no means the ultimate architecture pattern, the only one that can be effectively used to organize and manage IT assets.

A situation in which the CIO commands her team to “implement SOA” or asks for budget to support the “SOA rollout” should raise caution. On the other hand, if the CIO asks her architects to apply SOA to solve enterprise or application architecture problems, this is likely the right thing to do. SOA is probably worth adopting as the main guiding principle in organizations looking for ways to achieve interoperability of a large, heterogeneous portfolio of applications; simplify the access to diversified application functionality for business users; or provide the means to reuse functionality of existing

applications to support new or ad hoc business processes. Still, there are many other architecture “macro patterns” that are still helpful and perfectly valid for specific tasks.

Rearchitecting for SOA will most likely bring little gain for enterprise architectures dominated by a few large application packages that deliver most of the functionality required by the business. In such situations, it is much more important to track the evolution of package architectures and adjust the rest of the enterprise architecture for smooth integration with the core packages. In time this will eventually result in evolving to SOA, as many vendors — including SAP, Oracle, and Microsoft — are currently announcing significant R&D investments to make the architecture of their packages service oriented. This is an important trigger for the adoption of SOA, as the service-oriented packaged solutions will offer large chunks of functionality readily available as services, making SOA investments much more cost effective.

While any major architecture decisions are always very context specific, there are some rules that emerge as, if not “best,” then at least “good enough practices.” The roadmap we would like to present as a generalization and the main “takeaway” of the article is based on experiences from over a dozen projects involving SOA, EAI, and multichannel architectures (including the projects presented in the case studies). We believe that a

good strategic roadmap needs to clearly indicate its destination — a set of goals or a vision that (if adopted) gives meaning and direction to all the paths and intermediate milestones.

Adaptive Enterprise Architecture

In our SOA roadmap, the destination is called the Adaptive Enterprise Architecture. It is a vision of the enterprise in which the business processes, information, and knowledge assets can be quickly and effectively (re)organized and (re)deployed to support and enable the strategic maneuvers, innovations, and productivity levels required by today’s highly competitive business environment. The Adaptive Enterprise Architecture separates the business from the complexity and diversity of IT assets such as networks, servers, and applications in the same way that fly-by-wire systems separate pilots

from the complexity and diversity of the infrastructure required to maneuver modern planes.

The final destination of the SOA roadmap can be reached by taking relatively small steps. Each step, representing a stage in learning how to apply SOA, relates to different categories of architecture management issues and results in specific business benefits and opportunities. The steps included in the roadmap are: service delivery, service integration, service interoperability, and service management.

Service Delivery

The software service — a chunk of business logic with well-defined semantics and a well-defined interface — is the key building block of the Adaptive Enterprise Architecture. One cannot apply SOA principles in an IT environment where there are no services.

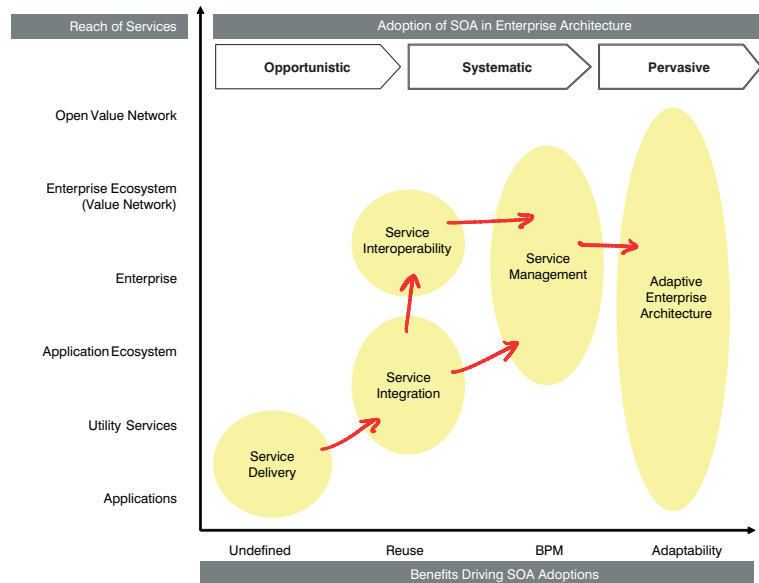


Figure 3 — The SOA roadmap.

Service delivery starts with applying SOA technical standards (such as Web services protocols, definition languages, middleware technologies) on the application level. This type of adoption is usually driven by specific application requirements, typical for platform solutions or niche utility services shared across the enterprise, such as security or authentication infrastructure. In the not so distant future, organizations will be able to procure services as part of packaged application suites. At this stage, services are just another way of delivering software functionality across a network, and the benefits of the investment in mastering the new paradigm are very much application specific.

SOA will thrive in domain areas where the service portfolio is already rich and the demand for new applications remains high.

Service Integration

To realize the broader benefits of SOA, it is important to change the focus of service implementation efforts from applications to the enterprise architecture. The first areas where the investment starts to pay off are the reuse of existing application functionality in new, composite applications and — if SOA is applied as the architecture pattern for enterprise application integration — the elimination of the

application integration spaghetti that is common in heterogeneous, highly interconnected application environments.

This step results in the assembly of service portfolios, defined and published using standard mechanisms such as UDDI directories. It also stimulates the organization to evaluate and select tools that can increase productivity in the delivery of new services and composite applications.

The value of service integration depends on the quality of services as much as on the quality of the entire service portfolio. To achieve the benefits of reuse, one has to pick the right functionality — functionality with significant reuse potential — and deliver it in a way that is both generic and practical across different application contexts.

Service Interoperability

The rising proficiency in using services leads to a situation in which, for some types of applications, SOA will gradually become the preferred way of delivering functionality. SOA will thrive in domain areas where the service portfolio is already rich and the demand for new applications remains high. Once the productivity tools are in place, assembling new business functionality from existing services will often prove to be cheaper, faster, and more reliable than other approaches. This will lead to the creation of application ecosystems

— sets of applications sharing significant portions of functionality.

The use of services outside such ecosystems or, indeed, sharing the services with business partners may sometimes prove difficult because of a lack of precise specification of service semantics and “modus operandi.” Unexpected behavior — often mitigated as a “feature” in the application ecosystem but unexpected outside its boundaries — will sometimes lead to interoperability problems.

To cope with such problems, organizations will need to employ a more disciplined approach to enterprise architecture management, creating service interoperability standards, extending service repositories, and employing service configuration management techniques. It will also be important to implement systematic service refactoring as part of these service portfolio management processes.

Service Management

The effective and efficient collaboration of services across many often unforeseen contexts creates demand for more elaborate definition of service capabilities. Apart from the interface definition, which is the fundamental SOA metadata, applications have to be able to determine capabilities such as performance, reliability, or cost of service.

Managed service portfolios are a prerequisite for the serious implementation of business process

management (BPM) tools based on SOA. While it is technically feasible to use BPM tools and approaches to assemble business processes once the service portfolio is in place (the service integration stage), one can only achieve the benefits of BPM if the portfolio is rich and reliable. Otherwise business operations may be in jeopardy.

At this stage it is critical to implement enterprise architecture management practices and tools that enable the maintenance of a broad range of service metadata, defining standards for extended service definition and service level negotiation protocols extending the functionality of service directories. Such protocols may be defined and automated using rule engines or implemented manually as “librarian” services to existing directories.

CONCLUSION

The steady, rapid growth of computing often redefines the impact computers have on strategy, making proven solutions obsolete and turning long-forgotten ideas into disruptive innovations. This is probably the most important reason why IT still represents an asset that has strategic consequences, especially in areas of enterprise agility, innovation, and efficiency.

It is therefore important to carefully evaluate the hype generated by vendors and industry analysts on the one hand and avoid resorting to the “we have all been here before” thinking on the other. That kind of thinking is sometimes employed as an escape from the effort and pain of systematic relearning and challenging the status quo in the business-technology relationship. There is no escape from learning and changing in a knowledge-based economy. We believe that SOA benefits should be carefully evaluated within every major IT organization, especially where application heterogeneity, knowledge productivity issues, high rate of change in business, and B2B relationships come together as the IT challenge. We hope that the short case studies and the roadmap we have provided here will help *Cutter IT Journal* readers resolve the SOA dilemma for themselves.

Bartek Kiepuszewski is a senior consultant specializing in J2EE architectures, workflow systems, and agile software development. Dr. Kiepuszewski holds a Ph.D. from Queensland University of Technology, where he worked on theoretical foundations for workflow modeling languages. Before joining Infovide in 2001, he worked for five years in Australia, first at the Distributed Systems Technology Centre as a research scientist, and later at Mincom Ltd. as a senior J2EE architect. Dr. Kiepuszewski has authored numerous research papers, primarily in

the workflow area, and he is a frequent speaker at national and international conferences, seminars, and trade shows.

Michał Paluśkiewicz is a senior consultant at Infovide, specializing in J2EE systems and agile software development. Since beginning his career at Infovide in 2000, Mr. Paluśkiewicz has participated in numerous J2EE projects, mainly as an architect, expert, or a team leader. He has authored many J2EE trainings and taught courses, during which he always tries to transfer to listeners as much of his hands-on experience as possible. Mr. Paluśkiewicz has written a number of papers and published articles on J2EE and mobile technologies.

Borys Stokalski is a cofounder and principal of Infovide, a Polish company specializing in enterprise architecture consulting and implementation. Infovide is ranked among the top IT consulting organizations operating on the Polish market.

Mr. Stokalski began his career in 1985 as a research worker and then as a lecturer in the area of human-computer interaction at Warsaw University. Today his primary responsibilities include involvement in business development, innovation, and strategy. He takes an active part in Infovide initiatives targeted at new service areas and consulting products. Mr. Stokalski occasionally takes the opportunity to perform high-level consulting and coaching for selected Infovide customers. He frequently publishes articles on enterprise architectures, business-IT alignment, IT strategy, and IT-based innovations, and is a frequent speaker at conferences and seminars.

The authors can be reached at Infovide S.A., Kolejowa 5/7, 01-217 Warszawa, Poland. Tel: +48 22 534 74 00; Fax: +48 22 534 74 02; E-mail: bkiepuszewski@infovide.pl, mpaluskievicz@infovide.pl, bstokalski@infovide.pl.