Diversity in Reuse Processes

Four European companies successfully achieved software reuse despite pursuing radically different processes and technologies. The authors explore the reasons behind the companies' success and present guidelines for others wanting to establish reuse programs.

Maurizio Morisio, *University of Maryland and Politecnico di Torino*
Colin Tully, *ESPI Foundation*
Michel Ezran, *Valtech*

Software project managers can confront seemingly schizophrenic conditions daily. Projects might be out of control, and processes and techniques can vary tremendously, while technology changes at a dizzying pace. On the other hand, the literature abounds in discussions of one-size-fits-all process frameworks, which presents managers with an apparent contradiction.

Process diversity can help us resolve this contradiction. It is a given, for example, that software engineering should apply defined, repeatable processes to achieve predictable results. However, processes vary from company to company and from project to project, so one approach clearly cannot serve all.

In this article, we examine the impact of process diversity in one specific area: software reuse, which means the work products developed elsewhere—in another project, group, or company—are used again. Typically, the work product is code, but reuse can be leveraged to an even greater degree if requirements, design, testing, and all related work products are involved.

In practical terms, how do we attain reuse? The literature suggests a certain commonality in approaches. For example, Jeffrey Poulin underlines the shift over the years from the library-based approach (in which a library contains many work products) to the focused-repository concept (in which a repository contains only a few, useful, and high-quality work products).[1]

To learn how to achieve reuse, we initiated a two-year study in 1997 of roughly two dozen European companies that were establishing reuse programs in the context of Process Improvement Experiments.[2,3] The sidebar, "Research Method," explains how the study was performed. We discovered that, despite diversity in business context, technical and managerial tradition, and size, companies can indeed achieve reuse, using diverse processes. From that study, we present four companies as case histories, selected for the variety and effectiveness of their approaches.

0740-7459/00/$10.00 © 2000 IEEE

Software Reuse

Software reuse takes two forms: *Ad hoc*, or *opportunistic, reuse* happens by chance, owing only to an individual's goodwill. Its usual form is cut-and-paste. *Systematic reuse* results from planning, executing, and monitoring changes in the processes at the organizational level to ensure regular reuse.

Reuse promises certain advantages, chiefly because new software need not be written. Cycle time shortens because fewer work products must be developed; quality improves because already produced work products are already debugged; and the combination of both reduces costs. Further, only one work product must be maintained instead of several.

However, although the concept is simple, the implementation is not. Achieving reuse of a work product requires satisfying at least three conditions:

- *Functional reusability*. The work product must offer functionality needed by the project.
- *Technical reusability*. The work product should be easy to integrate in the project's operating environment (such as a database.
- *Quality*. The work product quality level should meet project requirements.

To ensure the first amd second conditions, companies should establish processes and roles to identify the functionality likely to be needed by future projects and to capture it in reusable work products isolated as much as possible from the operating environment. Finally, to meet the third condition, companies need processes that qualify and document work products.

Research Method: Reuse Case Studies

We collected the case histories described in the main text through structured interviews concerning industrial reuse projects, in the framework of a European research effort. Our goal was to gather industrial reuse experience and present it in a series of practical guidelines to implement a corporate reuse program. We took these steps:

1. *Identification and selection of reuse projects*. The European Commission has funded several hundred Process Improvement Experiments. Each is a technology transfer project in which a company or organizational unit applies a specific software technology. In our study, this phase selected only PIEs dealing with reuse.
2. *Development of the questionnaire as support to the structured interview*. We developed the questionnaire in several iterations. We tested the latest versions with dry runs, then modified the questionnaire according to the feedback we received.
3. *Reading of reports and interviews*. We contacted the selected PIEs to schedule an interview at their premises. Before the interview, we thoroughly read the report and any other information available. One or two of us conducted the interview, which was usually made with the PIE project manager (who was guaranteed the anonymity of the data provided) and which lasted for several hours. The interviewee received a copy of the questionnaire; the interviewer read questions and noted down the answers.
4. *Validation of the interview report*. We produced an interview report and sent it to the interviewee for approval.
5. *Coding and consistency check*. We coded the interview report in a number of variables. Some variables had preformed codes, so actually coding was performed directly during the interviews. In other cases, we reorganized and changed the variables and assigned codes after the interviews (postformed codes).
6. *Data analysis*. We analyzed the data set to extract findings and trends.

Our questionnaire, data set, and analysis results are detailed elsewhere.[1,2] For more information on the case histories, see our forthcoming book *Practical Software Reuse*.[3]

References

1. M. Morisio, M. Ezran, and C. Tully, "Introducing Reuse in Companies: A Survey of European Experiences," *Proc. Fifth Symp. Software Reusability* (SSR '99), ACM Press, New York, 1999, pp. 3–9.
2. M. Morisio, M. Ezran, and C. Tully, "Success and Failure Factors in Software Reuse," http://morserv.polito.it/papers (current June 2000).
3. M. Ezran, M. Morisio, and C. Tully, *Practical Software Reuse: The Essential Guide*, to be published by Tata McGraw-Hill, New Delhi, India, 2000.

The Four Case Histories

Established about eight years ago, Sodalia is a relatively young SME (small-to-medium enterprise) specializing in telecommunications software. It employs a staff of 300, all dedicated to software production, and provides customized *telecom network management* systems. In TNM, products offer different levels of services and functions (such as network configuration management, performance management, and fault management) that are customized for each client. The company is ISO-9001 certified and has been assessed at CMM Level 3. Developers, who work in C++ from UML object-oriented designs, use frameworks to organize large collections of classes and Corba to support distribution.

Thomson-CSF is a multinational industrial conglomerate developing products and systems based on electronics and software. Of 50,000 Thomson-CSF employees, 5,000 work on software production. Although the conglomerate consists of several smaller companies or business units, our focus is on the

Table 1

Company Characteristics

Characteristic	Companies			
	Sodalia	**Thomson-CSF**	**Eliop**	**Chase**
Company organization	Medium/small company	Conglomerate, organized as business units (BUs)	Small company	Small company
Business context	Telecom network management	First BU: air traffic control; second BU: training and simulation systems	Industrial control systems	Mutual marine-insurance industry
Software staff (size)	300	5,000 at corporate level; 200 and 10 for the two BUs surveyed	30	27
Software process maturity	ISO 9001; CMM Level 3	ISO 9001; CMM Levels 2 and 3 for the BUs involved	ISO 9001	Informally self-assessed as CMM Level 1
Development approach	Object-oriented analysis and design (UML); C++ and Corba	Structured or object-oriented analysis and design; C or Ada	Structured analysis; C; assembly language	Object-oriented analysis and design; Powerbuilder and SQL
Type of software	Technical	Technical, embedded, real-time	Technical, embedded	Transactional

corporate reuse program and on two business units that apply it. One, a 10-developer unit, produces training simulation systems; the other unit, with 200 software staffers, produces air traffic control systems. The business units' process maturity levels vary. The two we consider are both ISO-9001 certified, and are at CMM Levels 3 and 2, respectively. The first unit's development approach is based on structured analysis and C; the other's approach is based on object-oriented analysis and Ada.

Eliop is an SME with 100 employees, 30 involved in software production. The company develops industrial control systems, such as remote telecontrol, process automation, and energy management. The software is embedded in in-house microprocessor-based boards and runs on standard computers. The systems control critical industrial processes and require real-time continuous operation. Software is a crucial value-added element of Eliop products. The company is ISO-9001 certified, and software developers use structured analysis, C, and various forms of assembly language.

Chase Information Technology Services ("Chase" hereafter), with 27 developers, produces end-user applications to support mutual marine-insurance processes. (At the time of our study, the company name was Chase Computer Services.) These applications, broadly similar but adapted to individual customers, are client-server, GUI-intensive systems. Developers use SQL, Powerbuilder, and object-oriented analysis. Chase is not ISO-9001 certified.

Table 1 summarizes the companies' general characteristics.

All four companies have a limited number of customers and produce software having similar functionalities that the companies adapt for each customer. This type of business contrasts with a company's producing off-the-shelf software and selling it to a huge customer base, or producing a series of individual but completely unrelated projects over time.

All companies except Sodalia dealt with customizing software for a relatively small customer base by regarding each new project as an upgrade, made of modifications and adaptations, of a previous project. Sodalia used the product line concept from the beginning.[4,5]

Comparing the Reuse Processes

At some point, the companies had decided they needed to improve their software process to address the reuse issue. All companies viewed the transition to reuse as a strategic one, which top management endorsed and which a senior executive directly supervised. The factors driving the companies to select reuse as the technology-transfer object invariably involved better productivity, faster time to market, and improved maintenance.

The companies intended the reuse initiative to determine requirements and code for a somewhat sophisticated yet generic and adaptable software product. Furthermore, the companies planned to introduce processes to support the maintenance and evolution of the generic software product.

Table 2 highlights the key characteristics of the four companies' reuse processes.

Reuse approach. As the table indicates, all four companies approach reuse differently. Thomson-CSF and Sodalia use an approach based on defining product lines.[4,5] A product line addresses one or several well-identified business subdomains. Usually a product line is built through domain engineering and application engineering. Domain engineering aims at defining and implementing domain commonalities in a generic product. Application engineering produces individual applications for customers starting from the generic product.

Eliop has performed domain analysis, an activity part of domain engineering, to develop reusable vertical assets, such as an invoice component in an accounting domain. The company has simultaneously tried to change from ad hoc to systematic reuse of horizontal assets—those suitable for any domain, such as GUI components. Moreover, Eliop has adopted *white-box reuse*, which lets the reuser modify the reused asset, because it entails less cost and risk. White-box reuse is also organizationally simpler because it does not require specific personnel assignments for reusable-asset development and maintenance.

Chase has shifted its development process from project to product oriented. Although not formalized, this approach appears similar to domain engineering.

Reuse technology. Reuse technology refers to the means for building and integrating the coding assets. Sodalia applies object-

Table 2

The Companies' Reuse Processes

Process characteristic	Companies			
	Sodalia	**Thomson-CSF**	**Eliop**	**Chase**
Reuse driver	Productivity; offering customers more functionalities	Reduced cost; faster time to market	Productivity; reliability	Exploit commonalities; reduce maintenance burden
Reuse approach	Product lines; analysis of commonality and variability	Product lines; analysis of commonality and variability; development of an application baseline (an adaptable, extendible core product)	Transform reuse of horizontal assets from ad hoc to systematic; introduce reuse of vertical assets, with limited analysis of commonality	From several development teams in charge of a customer to a team developing a product plus individuals in charge of customizations
Reuse technology	Frameworks; class libraries; components	Functions and modules	Functions and modules	Class libraries
Reuse roles	Reuse leader; reuse support engineer; reusable-assets developer; reusable-assets repository manager	Reuse leader (corporate level); reuse correspondent (in each BU); configuration manager (BU)	Reuse leader; library manager	Library manager; no other specific reuse roles, but all roles were modified with reuse responsibilities
Who develops assets	Reuse group	Project team	Project team	Project team
When assets are developed	In advance	When needed by the first reusing project	When needed by the first reusing project	When needed by the first reusing project
Reuse processes added	Domain engineering; reusable asset repository management; asset validation	Domain engineering	Domain analysis; asset validation	No specific reuse processes, but all nonreuse processes were modified
Nonreuse processes modified	All	Requirements; design; testing	Requirements; design; testing	All
Repository	Specific tool, developed in-house on top of a configuration management tool	Configuration management tool	Configuration management tool; database application for cataloging assets	Configuration management tool
Human aspects	No resistance to reuse; reuse started with the company and is accepted de facto by new employees; training and awareness initiatives	Almost no resistance to reuse, because of mature processes and strong management support; training and awareness initiatives	Some resistance to reuse; training and awareness initiatives	Some resistance to reuse; training and awareness initiatives

Putting a reuse approach and technology into practice requires that a company design a suitable process for doing so.

oriented technology, in the form of class libraries and frameworks. The company implements reusable assets as C++ and Corba frameworks, and it uses class libraries for domain-independent assets. It develops an application by combining, extending, and adapting these frameworks.

Chase uses object-oriented technology and class libraries. An earlier, separate requirements model created for each client has been replaced by a central requirements catalog, to which any new client-specific requirements are added. Each requirement is linked to sets of classes. In code development, Chase developers achieve reuse through a combination of direct reuse, inheritance, and parameterization.

Eliop and Thomson-CSF rely on procedural technology: functions and modules. Thomson-CSF uses the product baseline concept, a collection of C and Ada functions and modules that implement a generic, customizable application. Developers build each new application starting from the product baseline, then changing and adding functions as needed. Periodically, the product baseline is updated, and new functions from applications are integrated while others are deleted. Eliop's reusable assets are functions and modules, with the related documentation.

Reuse processes and roles. Putting a reuse approach and technology into practice requires that a company design a suitable process for doing so. In general a company must define new reuse roles and responsibilities, add reuse processes, modify nonreuse processes, and install tools—typically a repository for reusable assets. Roles and processes clarify who develops a company's reusable assets and when.

Sodalia has established a two-level reuse organization. A company-level reuse group is responsible for the following:

- reuse process definition and update;
- methodological support to projects;
- external asset acquisition;
- development and maintenance of an in-house tool to manage the asset repository;
- development of components common to all product lines;
- repository management;
- reuse coordination; and
- reuse measurement, tracking, and reporting.

The company-level team collaborates with the teams at the second level. The second reuse level focuses on product lines. Sodalia currently has three product lines defined. Each is implemented by an application framework and supported by a group (a product line manager and developers) for product development and maintenance. Sodalia assigns a project manager to each new customer project, with the application developed in one or more of the appropriate product lines.

At Thomson-CSF, a full-time corporate-level reuse group supervises reuse programs in each business unit and offers guidance and consulting. Each unit is free to choose a particular reuse approach and technology, but the top-level reuse group defines a specific reuse approach and technology. Although they are not required to do so, the business units normally do follow the defined approach.

The reuse leader, who is part of a corporate-level group charged with technologies and technology transfer, spearheads the reuse program. In the SME, the leader is usually the company owner or CEO; in a large company the leader can be the quality manager or come from within the R&D department. The reuse leader is responsible for

- promoting and disseminating reuse culture into the whole group,
- monitoring reuse adoption in all business units,
- coordinating reuse activities throughout the group, and
- tracking and consolidating reuse results from all business units.

Because business units are independent profit centers, each unit is free to "subscribe" to the reuse program or not, and, if it does, to use its own technologies, approach, and repository. Each business unit has its own reuse program, reuse organization, and software assets, which are not shared. Business units involved in reuse have, at least, a reuse correspondent, an individual who manages the reuse program in the unit and reports to the corporate level.

The two Thomson-CSF business units in our study appointed a part-time reuse leader and do not separate reusable asset production and use. Each project maintains and produces the reusable assets on a part-time basis. So, reusable assets are produced just in time by

the first reusing project. The configuration manager is the logical person for managing reusable assets, because they are stored in the configuration management tool. The business units added a domain engineering process, which was supported at the corporate level.

Eliop and Chase essentially follow the same organization as Thomson-CSF but without the corporate reuse group. At Eliop, developers are organized into teams, with one or more teams for each software environment. The company introduced the following roles: reuse leader, with overall responsibility for reuse processes; reuse task group, consisting of the reuse leader, two team leaders, and two senior engineers, responsible for domain analysis and asset validation; and a library administrator.

Chase has a flat organizational structure. All senior managers have multiple roles; therefore, organizing the introduction of reuse as a separately staffed "insulated" project was not practicable: Staff were assigned specific reuse tasks from time to time, in parallel with their mainstream work on development projects. Before the reuse project began, Chase development teams were organized by client. Once the project began, teams were organized by each phase of the reuse process, such that each team worked with all clients. This organization fosters a better understanding of the reusable assets appropriate to each client and to each phase. A manager responsible for each client provides continuity between phases.

Before Chase's reuse project, the development process for each new customer began with an existing system, which developers then cloned and modified as necessary. Unfortunately, this approach eliminated traceability between different customers' requirements, making it hard for Chase to capitalize on any application similarities. Moreover, maintaining this growing number of different, though similar, systems became almost unmanageable. In the revised reuse process, each project generates defined deliverables, compliant with defined standards, at key milestone points and as much as possible from a catalog of existing assets. Assets are not maintained by direct editing but indirectly through base-asset maintenance.

Sodalia and Thomson-CSF use domain analysis and domain engineering extensively. Eliop uses domain analysis on a limited basis; Chase does not use it.

In all four companies, again, the repository is implemented on top of the existing configuration management tool, with or without extension through reuse-oriented functionalities. In all cases, except for Sodalia, the configuration manager supervises the reusable assets and takes the role of library manager.

Human aspects. Somewhat surprisingly, these four companies experienced no serious problems with human aspects. This is likely due to a strong management presence. Management, from the start, understood the importance of reuse and determined a strategy based on awareness actions (such as presentations, briefings, and newsletters) and training plans to address and anticipate problems.

Results. To assess the results of these reuse programs, we applied a pragmatic definition of success, measured against three criteria: the reuse program continues after the 18- to 24-month Process Improvement Experiment completes; reusable assets are routinely reused; and the reuse approach is judged as sound. (Ideally, a return-on-investment analysis should be performed to formally assess a reuse program's success, but the necessary supporting data is seldom available.)

The companies satisfied all three criteria and can thus be declared successes. This is not necessarily typical; roughly one-third of the other companies we surveyed abandoned the reuse program because of poor results or an inability to make it work.[2,3]

Furthermore, we can compare the results achieved against the initial reuse drivers. Sodalia, for example, which wanted increased productivity and more extended, flexible product offerings, believes that the reuse policy contributes to these. The company is now organized around the reuse approach; moreover, the program continues to grow. Its repository contains approximately 100 large-grained valuable assets that capture business and technical expertise. Sodalia has used existing assets from one to five times. Costs are controlled because the reuse team has an established budget.

At Thomson-CSF, the drivers were cost and, especially, time to market. The two business units' market is mature and highly competitive, so time to market is key to

Somewhat surprisingly, these four companies experienced no serious problems with human aspects.

This maturity significantly helps companies understand and control their processes and determine the direction in which the processes should evolve.

maintaining or increasing their market share. For one business unit, average time to market has dropped from two years (in 1993) to 11 months (as of 1997). Significant productivity improvements have let the units reduce costs and become more competitive. The reuse level—percentage of reused code in a new system—can exceed 75% when product lines are well mastered.

At Eliop, the drivers were productivity and reliability. Reuse is now part of the company's regular software development practice; the repository contains approximately 70 code assets and 50 document assets. Most coding assets have been used at least once—several, more than once. We estimate that in this context, savings result with the second use. Large productivity improvements are unlikely. Reliability has improved, however, and Eliop considers this benefit even more important.

Chase's main driver was the reduction of both rework and corrective maintenance. Reuse is now central to the company's development approach, incorporated in all phases of reuse with built-in review points to ensure that developers consider all reuse possibilities. The results are better time to market, reduced demand on scarce resources (particularly through reductions in rework), product quality, and client confidence.

Keys to Success

Six key points underlie these four deeply diverse success stories:

- *Change processes and roles*. In different ways, every company has introduced reuse processes and roles, and they have modified nonreuse processes, especially those involving requirements and design.
- *Obtain management commitment*. To change processes and roles requires knowing what they are and being able to modify them. The first point requires knowledge of the company and the state of the practice. The second requires management support to obtain resources, and the power and will to modify existing practices. All four companies had good knowledge of the existing processes, because of either their process maturity or their limited size. They also could count on strong management commitment to enact the changes.
- *Minimize changes*. The companies retained their existing development approach and chose reuse technology to fit that approach. They used their existing configuration management tool for the repository, although Sodalia added new functions. The advantage here is in introducing as few changes at a time as possible and in building on existing knowledge, skills, and tools. The central question becomes, what is worth changing and what is not? The four case studies teach us that change should focus primarily on processes and roles. Development technology and supporting tools can be changed later, if necessary.
- *Keep a sense of proportion*. Changes to processes and roles should be affordable. The companies' choices varied greatly, yet if we contrast them with a company's size and available resources, a logic appears. Sodalia is the biggest company per se (disregarding Thomson-CSF's conglomerate status in favor of only two business units), and it can sustain a separate reuse group, whereas the others' small size discourages such a group. The same applies to domain engineering, a process that only Sodalia and Thomson-CSF can afford (the Thomson-CSF corporate level, not individual business units, sponsors the process). Again, assets are developed in advance by the biggest company, while the others develop assets just in time for the first reuse.
- *Anticipate human factors*. In all cases resistance to the initiative was low, not by chance but because training and awareness activities were planned in advance.
- *Acquire process maturity*. Most of the companies had solid process maturity before beginning their reuse initiatives. This maturity significantly helps companies understand and control their processes and determine the direction in which the processes should evolve.

The lessons learned from our case histories can be expressed as a series of straightforward guidelines for companies wanting to define their own reuse program.

First, evaluate the *reuse potential*: How many similar software products or projects are produced over time? A stable and well-

defined business context is usually a prerequisite for reuse potential.

Second, if reuse potential is high, a reuse program can be started, provided *reuse capability* is achieved. Initially, get top management's commitment to acquire resources, and get the power to

- change nonreuse processes,
- add reuse processes,
- address human factors, and
- set up a repository.

Adding reuse processes normally implies defining and assigning key reuse roles, so the latter is listed implicitly.

Be sure to check ownership of processes and requirements. Changing nonreuse processes and adding reuse processes will be much more difficult when ownership of these processes lies elsewhere—that is, when subcontracting is involved.

Finally, *reuse implementation* can start. Each of the above issues involves these lower-level choices and actions.

- *Change nonreuse processes*. Requirement definition and analysis, high-level design, and testing all require specific changes that take into account asset availability. Project management is also impacted, especially concerning scheduling, costs, and productivity.
- *Add reuse processes*. Domain analysis can drive the identification of reusable assets. Assets can be small or large, including design and requirements. They can be developed from scratch or reengineered from legacy. They can be produced and maintained by a specific group or by projects, before projects need them or just in time for the first use.
- *Address human factors*. One or more techniques (such as training, awareness events, discussion groups, and newsletters) can be used. Incentives alone are not sufficient.
- *Set up a repository and support it with a tool*. A specific tool, add-ons to the configuration management system, or the plain configuration management system are all possibilities.

The availability of resources in the company, usually related to its size, should be carefully considered to make sustainable choices. Provided the approach is sustainable, integrated, and adapted to your individual context, any combination of choices is acceptable.

Acknowledgments

We thank the European Commission for funding the project (Surprise, Esprit/ESSI project 23960) and specifically the project officers Andrea Di Maio and Gisèle Roesems. We also thank the four companies for their active and much-appreciated help; in particular, Giovanni Cortese of Sodalia, Pascal Maheut of Thomson-CSF, Manoel Villalba of Eliop, and Colin Woodgate of Chase Information Technology Services. The reuse introduction projects in these companies were partially funded by the European Commission under contracts Esprit/ESSI 21534, 21649, 10936, and 21826.

References

1. J. Poulin, "Reuse: "Been There, Done That," *Comm. ACM*, Vol. 42, No. 5, May 1999, pp. 98–100.
2. M. Morisio, M. Ezran, and C. Tully, "Introducing Reuse in Companies: A Survey of European Experiences, *Proc. Fifth Symp. Software Reusability* (SSR '99), ACM Press, New York, 1999, pp. 3–9.
3. M. Morisio, M. Ezran, and C. Tully, "Success and Failure Factors in Software Reuse," http://morserv.polito.it/papers (current June 2000).
4. M. Griss, P. Jonsson, and I. Jacobson, *Software Reuse*, Addison Wesley Longman, Reading, Mass., 1997.
5. J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," *IEEE Software*, Vol. 15, No. 6, Dec. 1998, pp. 37–45.

About the Authors

Maurizio Morisio is a research assistant at Politecnico di Torino, Turin, Italy. Currently he is on sabbatical leave at the University of Maryland, College Park, where he works with the Experimental Software Engineering Group. His research interests include experimental software engineering, software reuse metrics and models, the Personal Software Process, COTS processes and COTS integration, and software product line engineering. He consults on improving software production through technology and processes. Morisio received a PhD in software engineering and an MSc in electronic engineering from the Politecnico di Torino. Contact him at Politecnico di Torino, Dip. Automatica e Informatica, Corso Duca degli Abruzzi 24, 10129 Torino, Italy; morisio@polito.it; http://morserv.polito.it/morisio.

Colin Tully is director of the European Software Process Improvement Foundation. Since 1989, as principal of Colin Tully Associates, he has been an independent consulting engineer, specializing in systems and software capability improvement. He has been extensively involved in the Esprit and ESSI programs, as project participant and reviewer, and in evaluating project proposals. He is editor in chief of the *Software Process Improvement and Practice Journal*. Contact him at the ESPI Foundation, 2 North House, Bond Ave., Milton Keynes, MK1 1SW, UK; colint@espi.co.uk.

Michel Ezran is chief knowledge officer for Valtech, an international consulting group. He is responsible for the management of knowledge and competencies across all North American and European operations and for the development of the company's intranet. Previously, as a senior consultant and R&D department head at Valtech, he advised companies on the migration of their IT infrastructures—systems, organization, processes, skills—to new technologies. Before joining Valtech, he worked for various service companies; notably, Cap Gemini in France and in South America. Ezran is a graduate of Supélec in France. Contact him at Valtech, Immeuble Lavoisier, 4, Place des Vosges, 92400 Courbevoie, France; me@valtech.fr.