Oil & Gas Science and Technology – Rev. IFP, Vol. 60 (2005), No. 4, pp. 587-596
Copyright © 2005, Institut français du pétrole

Software Interoperability for Petroleum Applications

B. Braunschweig[1]

1 Institut français du pétrole, 1 et 4, avenue de Bois-Préau, 92852 Rueil-Malmaison Cedex - France
e-mail: bertrand.braunschweig@ifp.fr

Résumé — Interopérabilité logicielle : les applications dans l'industrie pétrolière — Dans cette introduction au dossier sur l'interopérabilité logicielle, nous soulignons le rôle principal du logiciel pour l'industrie pétrolière, et nous présentons les motivations, aussi bien de marché que techniques, concernant l'interopérabilité. Ceci est suivi d'un rapide survol des technologies utilisées dans la recherche de l'interopérabilité, puis par les résumés des cinq contributions techniques qui constituent le cœur de ce dossier. Nous concluons en fournissant quelques idées sur les futurs défis d'interopérabilité dans notre industrie.

***Abstract* — *Software Interoperability for Petroleum Applications* —** *In this introduction to the thematic dossier on software interoperability, we emphasise the key role of software for the petroleum industry, and we give market and technical motivations for interoperability. This is followed by a brief overview of the technologies used in the search for interoperability and by a summary of the five technical contributions which constitute the core material in this issue. We conclude by providing some views on the future interoperability challenges in our industry.*

1 THE KEY ROLE OF MODERN SOFTWARE TECHNOLOGIES FOR THE PETROLEUM INDUSTRY

There is little need to emphasize the key role of computers and software for the oil and gas industries. Since the early days of computing, information technologies have been used by petroleum companies for every aspect of their business. Modelling and simulation have been central tools for technical and economical evaluation of assets such as exploration prospects, reservoirs, offshore equipment and transportation systems, refineries and petrochemical plants. The President of a large R&D company for the petroleum industry wrote in 2001 that information technology (IT) *"plays a major transverse role for us. In R&D, most research projects make use of modelling technologies and lead to new software. IT is a central element of our business, and contributes to establishing links between various sectors"* (translated from Mandil, 2000).

In the following, we emphasise a few characteristics of contemporary IT that must be taken into account when developing technical software for the petroleum industry.

1.1 Increasing Complexity

An important aspect of software in the oil and gas industry lies in its ever-increasing complexity. It is now well known that software artefacts are among the most complex technical systems built by mankind. Operating a refining complex, processing 3D seismic data, assessing the economic interest of an oil reservoir, all these tasks involve the combination of dozens of different software tools each of them providing only a part of the global functionality required. As an example, the Figure 1 shows the main technical components of a process modelling environment, as defined by the CAPE-OPEN standard (see article by Banks *et al., this issue*). Such an environment involves many technical functions, *e.g.* flowsheet management, management and calculation of physical properties, equation solving, optimisation etc., not counting the many user-oriented functions such as GUI(1), session management, saving and restoring data etc. The same can be said for other application environments *i.e.* reservoir simulators, drilling planners, etc.

1.2 Networks, the Internet and Mobility

Nowadays computer systems are networked, distributed, and mobile. Beyond the old paradigm of central computing systems, and beyond the more recent paradigm of personal computing, our current systems heavily rely on the availability of networked resources such as databases, workstations, supercomputing facilities, clusters, and of course the internet. The mobile oil and gas industry worker expects to find these resources at his/her fingertips, accessible from anywhere in the world, in the office, at home, on a rig, or in an airport waiting for a flight.

1.3 Evolution of Technologies

The software industry has developed new technologies at a very fast pace. Far from the initial monolithic systems of the

(1) Graphical User Interface.

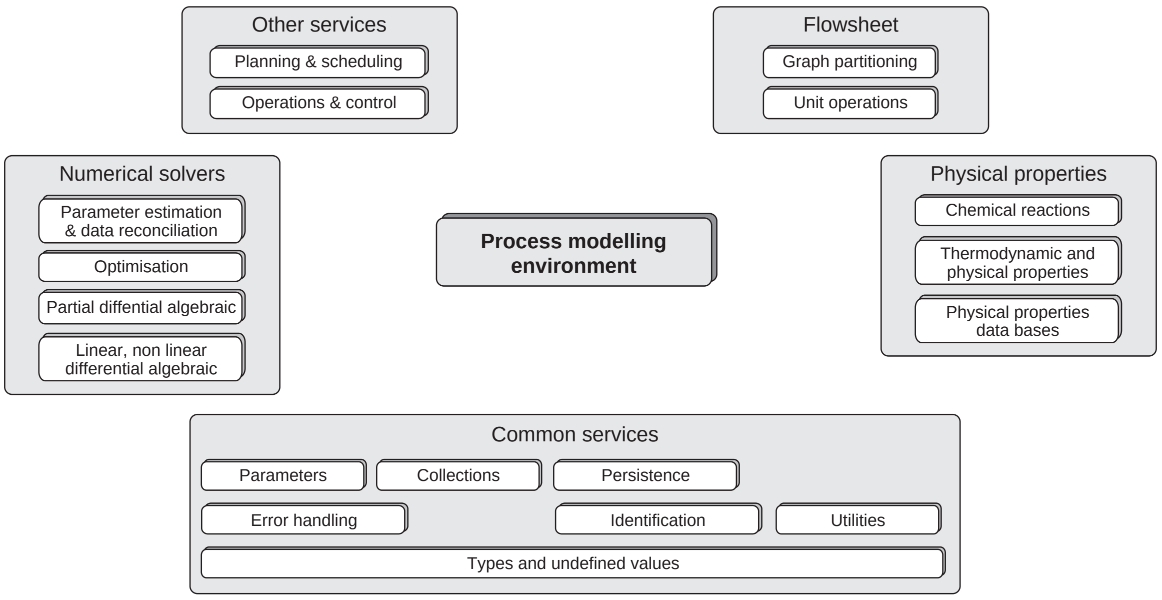

Figure 1

Architecture of a process modelling environment.

1950's and 1960's, software technologies reached a degree of sophistication in order to deliver the functions required by these complex distributed applications. Among these, object orientation, component technologies, web services and meta-object approaches are those which contribute most to software interoperability challenges, our the subject for this issue of *Oil & Gas Science and Technology - Revue de l'IFP*. With these tools, it becomes possible to compose a software system by assembling heterogeneous components, information and data.

1.4 The Need for Interoperability

Bluntly stated, no single software suite can do it all. Even the leading and larger worldwide software companies cannot supply the functions needed by all potential users of their tools and packages. Application needs are too numerous and too diverse, and sometimes unpredictable at the time when the software is developed. In a recent communication, Roger Sessions (Sessions, 2004), wrote about the motivations for interoperability in IT:

"The challenge today is not building enterprise software systems. Enterprise software systems are essentially commodity technologies. The challenge is building enterprise systems that can work together in innovative ways to create and exploit business opportunities. The key challenge today is interoperability.

Interoperability has a direct impact on profitability. Interoperability lowers costs by minimizing vendor dependencies, controlling failures, and reducing the cost of replacing antiquated systems. Interoperability increases revenue by allowing system synergies to be exploited, more cost effective technologies to be utilized, and expensive human resources to be maximized.

Enterprises that fail to embrace interoperability will simply cease to be competitive.

Systems interoperability starts at the highest levels of management where the value proposition for interoperability must be understood, analyzed, and embraced. This becomes the foundation for a corporate culture of collaboration that eventually pervades every aspect of corporate software development."

Although this text was written with corporate IT systems in mind, we believe that the same applies quite well to scientific and technical software applications for the oil and gas industries. These industries have triggered major software and data interoperability initiatives such as POSC or CAPE-OPEN in the last ten or fifteen years, spending millions of euros and US dollars to establish a technical basis for successful exchange of functionality and data at runtime.

The benefits of interoperability relying on standards for data and function exchange have been mentioned by several authors *e.g.* by OpenSpirit's chief technology officer, Clay Harter (Harter, 2004): *"With application integration middleware transparently connecting end users to the data that they need, and in the appropriate formats, valuable time and resources are shifting from data manipulation tasks to analysis and decision-making tasks—a shift that has compelling business benefits"*. Another significant example —and a sign of industrial interest—is the US Federal Trade Commission proposed consent order regarding the divestiture of Hyprotech simulation software from *Aspen Technology*, asking the latter to maintain compliance with open standards for several years in order to ensure a non-monopolistic situation in the process simulation market (FTC, 2004).

More concretely, the main benefits expected from software interoperability in our industry are:

– Easy, cost-efficient and error-free data transfer between applications; both in upstream and downstream applications, professionals need to transfer complex data structures from one application to another. Interoperability standards provide the mechanisms for fully automated data sharing and transfer between software from different suppliers. Errors are eliminated and productivity is significantly increased. Examples are: transfer of data from basin software to reservoir simulation systems, well engineering tools, etc., transfer of process flowsheets from process design systems to online optimisation and monitoring packages.

– Easy addition of functionality in application suites; open standards for runtime functional interoperability allow the seamless integration of external modules in software suites, in a plug-in fashion called "plug-and-play". Applications can then immediately benefit from these external functions as easily as using the same spelling checker in most popular office tools. Examples are: introduction of a specialised well steering algorithm in an interactive drilling application; introduction of specific thermodynamic calculations in process simulation packages.

– Avoidance of "vendor lock-in". Companies want to be able to move from one vendor to another while keeping and reusing their valuable assets in the form of data and models. Interoperability standards facilitate the process by giving vendor independence to the data and models.

– Trigger for innovation; when every single software company needs to provide every possible functionality to the user, a high amount of very valuable technical resources are lost in developing well-know systems that could be simply taken from others. With interoperability standards, these resources are employed for their best usage, that is, providing the special and innovative functionality that others can't provide. One example is that a process licensing company would just develop its proprietary reactor models, while relying on other companies' models for conventional unit operations.

– Support to quality assurance (QA) processes. When it comes to quality assurance, there is an obvious need to master all stages involved in a technical workflow. When some stages (*i.e.* data transfer) are automated thanks to software interoperability, QA can concentrate on the rest of the workflow without needing to worry about the possible mistakes made by human operators. Moreover, the utilisation of the same technical module (*e.g.* a corporate physical properties calculation package) in many environments is a key quality factor as it ensures consistency between applications and repeatability of numerical experiments.

We hope that the series of articles in this issue will show these benefits, and will present the breadth and relevance of interoperability projects and the potential of underlying technologies employed for their purpose.

1.5 A Market of Significant Size

The market of scientific and technical software in the petroleum industry is constantly monitored by consulting firms, therefore good estimates are available. There is data for upstream (*i.e.* geology, geophysics, reservoir engineering, drilling, production facilities) and for downstream (*i.e.* process design, operation, optimisation, monitoring and control, and operators training) applications.

In upstream applications, the worldwide market for scientific and technical software is generally estimated to be over 800 M€ and growing, with more than 50 000 workstations utilised by exploration and production (E&P) professionals equally distributed between geology, geophysics, reservoir engineering and drilling engineering.

In downstream applications, the overall worldwide market size is of the same order of magnitude, with around 300 M€ for offline process simulation and optimisation, and approximately 500 M€ for online systems and operators training.

Such markets have generated a reasonably sized software industry, with several large vendors providing integrated series of tools addressing a significant part of the scope. Both in E&P and in refining, a small number of major companies (*e.g. Schlumberger, Landmark, Aspen Technology, Honeywell, Invensys*) share more than two thirds of the market, whereas a large number of small-sized vendors provide other software suites with similar or additional functionality. There are also many *niche* vendors bringing added-value complements to the existing software base for specific applications.

1.6 Co-Opetition as a New Economic Framework

Software supply has been traditionally viewed as any other supply, subject to the market laws of competition. A typical customer would look at different options, do some kind of comparison based on technical and economical criteria, and decide to buy one tool or suite of tools from a single supplier. However, this approach, which has the advantage of being simple, becomes less and less relevant when it comes to setting up complex software suites made of interoperable components, since the final system will be composed of several pieces acquired from various sources. Economists have developed a new model for this "economy of interoperability", namely the "theory of co-opetition" as defined by Brandenburger and Nalebuff (Brandenburger and Nalebuff, 1996): *"Business is co-operation when it comes to creating a pie and competition when it comes to dividing it up. In other words, business is War and Peace. But it's not Tolstoy-endless cycles of war followed by peace followed by war. It's simultaneously war and peace. As Ray Noorda, founder of the networking software company Novell, explains: 'You have to compete and co-operate at the same time'. The combination makes for a more dynamic relationship than the words "competition" and "co-operation" suggest individually."*

In the petroleum industry, as for other industries, and following the trends of co-opetition, plug-and-play interoperability will stimulate the market and create new opportunities that could never have happened before. New value nets will be created with one supplier being another supplier's competitor, and at the same time the supplier's complementor, as assembling components from these two sources will provide more than just summing up the two parts by operating them separately.

2 A BREADTH OF SOFTWARE PARADIGMS AND TECHNIQUES

The first software developed in the sixties and seventies were large monolithic systems. Developed in FORTRAN, they were designed as large multipurpose programs sharing data through COMMON declarations and using internal or external subroutines. Modular programming helped in facilitating maintenance and debugging, so it was quickly adopted. At the core of these systems is the idea that they are self-standing, can solve by themselves all the problems, and will be extended by modifying source code, recompiling and deployment of the newly generated versions toward its users. Functional interoperability (*i.e.* the exchange of functionality between software at runtime) was just not an issue, since there was no practical way to implement it. Data interoperability (*i.e.* sharing of data between applications) was obtained by agreeing on common data structures and formats, and by passing one program's formatted output as input to another program.

Figure 2 illustrates a simple monolithic system for displaying a graph chart on a monitor: all modules depend on the shared memory area containing common definitions and variables *i.e.* colours, scales, data values, etc.

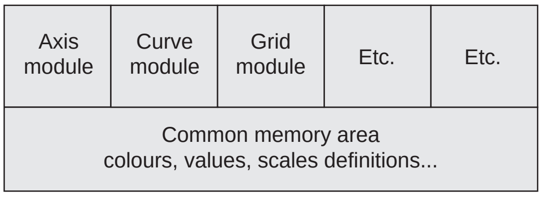

Figure 2

Monolithic system.

Object oriented ("OO") programming was introduced in the eighties and was progressively adopted since it constituted a better way to design, develop and maintain complex codes. In an object oriented system, software objects are instances of carefully designed classes; these classes define the generic behaviour of their instances through private and public properties and methods, and they exchange information and data through messages. This approach brings valuable properties such as:

– generalisation, a way to minimise software development by implementing functionality at the most appropriate level;
– encapsulation, a way to control access to the internals of an entity by authorising only predefined calls;
– polymorphism, a way to dynamically adapt the behaviour of an entity based on its characteristics, on its belonging to a specific class within the class hierarchy.

Figure 3 shows the same graph displaying modules in an object oriented fashion, where the axis, cure and grid objects expose their functionality to others through public methods while keeping their internal details in the private part, and communicate using messages.

The object oriented approach was widely adopted by the IT systems community because of its obvious benefits; however it took many years to develop in the scientific and technological software arena, as is was felt too complicated for inexperienced programmers, and less efficient in terms of performance. We can state that the OO approach was really adopted by the oil and gas industry in the nineties, with major projects undertaken by providers and users, thanks to the availability of good quality development and execution environments for C++ and FORTRAN 90. Good quality OO methodologies and formalisms such as UML (Object Management Group, 2004) were also key to the success of the OO approaches. These were only made available recently *i.e.* in the second part of the 1990's.

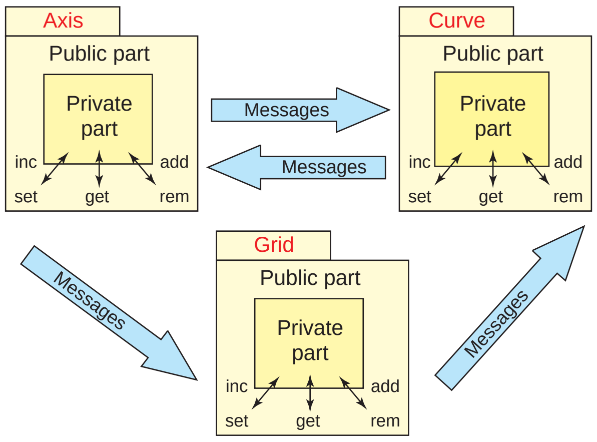

Figure 3

Object oriented software.

Object orientation is one possible technical solution to interoperability because of the way it forces to structure things. However, it is quite possible to develop un-interoperable object-oriented programs, and it is also possible —although less common—to develop interoperable software without an OO approach. But interoperating "conventional" OO systems still implies some access to the source code for adding or modifying functionality, or linking to other systems. Component technology provides the technical solution for interoperation without access to source code.

There is no commonly agreed definition of software components. *Webopedia* (www.webopedia.com) defines a software component as *"a small binary object or program that performs a specific function and is designed in such a way to easily operate with other components and applications"*. *Microsoft* presents its component model, COM, as *"the most widely used component software model in the world. It provides the richest set of integrated services, the widest choice of easy-to-use tools, and the largest set of available applications. In addition, it provides the only currently viable market for reusable, off-the-shelf, client and server components"* (www.microsoft.com/com/about.asp). The *Object Management Group (OMG)* presents CORBA as "OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network" (www.omg.org/gettingstarted/corbafaq.htm). *Sun Corp.* presents its *Enterprise JavaBeans (EJB)* component technology as *"the server-side component architecture for the Java 2 Platform, Enterprise Edition (J2EE) platform.* EJB *technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology"*.

Common to all these approaches is the notion of **interfaces**. Software components expose their functionality through published lists of methods and arguments, grouped in interfaces, that other software can use. These can also be seen as contracts between a software client and a

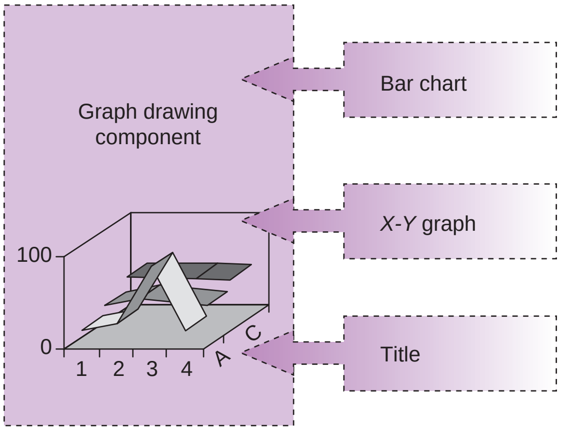

Figure 4

Graph drawing component.

server component providing a service. Figure 4 shows a graph drawing component exposing its functionality through interfaces.

Component technology allows programs to interoperate independently of their implementation. Source code access is not needed anymore, thanks to the contract-based behaviour provided by the set of interfaces. However, interoperating networked components developed in different languages, running on different hardware and operating systems implies the use of an intermediate software layer called *middleware*. Figure 5 shows the role of middleware between a graph drawing component operating on a PC, and a properties calculation components running on another computer and making calls to the graph drawing component for using its services.

Middleware comes in different flavours… unfortunately the software industry has not yet reached an agreement on this matter and many middleware solutions exist for interoperating distributed software components. We list some of the solutions here, noting that much more material is given in Belaud *et al.*'s article in this issue.

This issue contains examples of use of middleware for interoperability in upstream and downstream oil and gas applications. The content is presented below.

TABLE 1

Examples of middleware technologies

Middleware name	What it is, what it does
COM, DCOM, COM+, OLE, ActiveX, DNA	Microsoft's original middleware and architecture for Windows-based software components. Now superseded by .NET.
CORBA (Common Object Request Broker Architecture)	Object Management Group (OMG) middleware and architecture for distributed objects, using the IIOP standard protocol. Now integrated in OMG's CORBA Component Model.
XML (eXtensible Markup Language)	A metalanguage for designing specific markup languages for different types of documents and data. XML is used in several interoperability architectures.
.NET	Microsoft's current web-enabled development platform and interoperability framework, based on XML, SOAP and Web Services.
RMI (Remote Method Invocation)	Sun's technology allowing Java objects running on different Java Virtual Machines to interoperate.
EJB (Enterprise Java Beans)	Sun's architecture for interoperable distributed enterprise-level applications; usually dedicated to Java-based objects and components.
MOM (Message-Oriented Middleware)	A client-server infrastructure allowing distributed heterogeneous applications to interoperate thanks to publish-and-subscribe messaging mechanisms.
SOAP, UDDI and Web services	A wealth of XML-based technologies for interoperating software services over the Internet, based on SOAP as a communication protocol and on UDDI directories (yellow pages, etc.).
GRID middleware (*e.g.* Globus)	Technologies enabling the sharing of computer power and databases over the internet. The Globus Alliance provides the most popular middleware for this purpose.
FIPA (Foundation for Intelligent Physical Agents)	Non-profit organisation producing standards for the interoperation of heterogeneous software agents. FIPA provides several technologies including the Agent Communication Language (FIPA ACL).

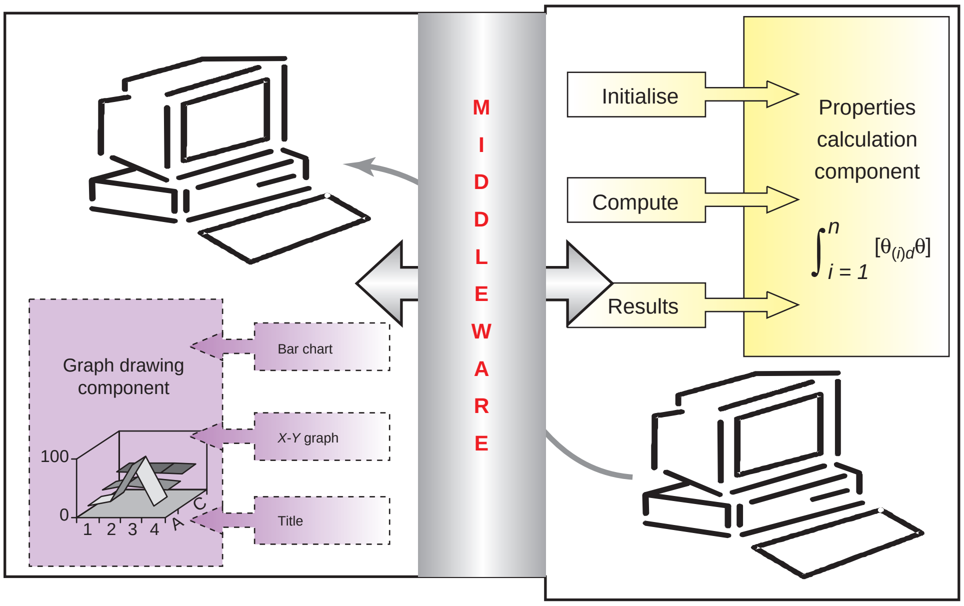

Figure 5

The role of middleware.

3 CONTENTS OF THIS ISSUE

This issue contains five technical articles. The first four show various aspects of software interoperability in five upstream and downstream technical application domains. The last article gives an overview of the software technologies used for interoperability and can be read last, first, or even by small sections when needed *e.g.* when the reader needs to understand the underlying technological aspects of a particular interoperability framework presented in the other articles.

- The paper by Rainaud gives a synthetic view of several interoperability projects dealing with underground aspects (geology, geophysics, reservoir). The oil and gas profession started to address underground data interoperability problems back in the 1990's with the well-known POSC consortium, which led to the reference Epicentre model. Since those days, many projects were undertaken, dealing with all sorts of applications (data exchange, software interoperability, workflow modelling, etc.) and using all sorts of technologies, from conventional database technologies to more advanced semantic models on top of XML. Rainaud's article will help the reader find his/her way among the POSC, Epicentre, RESCUE, OpenSpirit, EpiSEM, etc., projects for this key application domain. This view is complemented by looking at the very specific need of geo-referencing, that is, referencing objects of interest with geographical information. Born around ten years ago, the Open GIS consortium describes its vision as "*a world in which everyone benefits from geographic information and services made available across any network, application, or platform*". For this purpose, Open GIS supports open interface specification for spatial data, allowing interoperability of such spatial information. The article shows what these specifications are and what benefits they bring to our profession.

- The article by Banks *et al.* presents the most advanced interoperability standard in the process simulation field, that is, the CAPE-OPEN standard, the result of two successive EU-funded research projects gathering users, software providers and researchers in the domain of Computer-Aided Process Engineering (CAPE). The CAPE-OPEN technology has been extensively presented elsewhere (Braunschweig, 2002; Braunschweig *et al.*, 1999; Braunschweig and Gani, 2002; Braunschweig *et al.*, 2001), and is fully described on CAPE-OPEN Laboratory Network's (CO-LaN) web site (Co-LaN, 2002), therefore Banks and his colleagues essentially present a users view showing the main benefits that can be gained from interoperability of process modelling software. The reader interested in knowing more technical details should download reference articles from the CO-LaN's website.

- The next article, by Leal, shows a different view of interoperability standards as it is concerned with the seamless exchange of complex process/product data between process engineering applications. The need to exchange such process and product data has been identified a long time ago. The lifecycle of a plant or of a process imposes transfering complex data (material, fluid, equipment, operating conditions, etc.) along many phases *i.e.* conceptual design, detailed design, construction, start-up and commissioning, operation, maintenance and demolition. Initially powered by the STEP file standard, several projects have addressed this need, leading to the current implementations which take advantage of the XML format.

– The last application based article, by Matania, presents two complementary technologies for sharing application functionality online and in real time in refineries and petrochemical plants:
  • the well known OPC (OLE for Process Control) standard, developed on top of Microsoft's COM middleware, which allows interoperation of process control applications from several DCS manufacturers;
  • a framework for exchanging sophisticated process monitoring and supervision functionality (trends analysis, alarms processing, planning and scheduling, etc.) developed in the EU-funded CHEM project is presented.

  This framework makes use of publish and subscribe mechanisms with message-oriented middleware, and a first implementation was done using *Gensym's* G2 software, a commercial platform supporting knowledge-based process surveillance applications.

– The article by Belaud *et al.* presents the software technologies needed for interoperability: middleware such as Microsoft's COM and .NET, Object Management Group's CORBA, Sun System's Enterprise Java Beans, the widely accepted UML language for object-oriented modelling, plus other important technologies such as web services and Service-Oriented Architectures, etc. It provides a useful packed introduction to the domain, and pointers for those readers interested to know more about it.

Let us now conclude by providing some perspectives on further interoperability technologies making use of semantics.

4 PERSPECTIVES: SEMANTIC INTEROPERABILITY, SEMANTIC WEB SERVICES

We believe, as many others do, that the next stage of software interoperability will rely on semantic web technology using domain ontologies and web services frameworks. The current worldwide web is very rich in terms of content, but is essentially syntactic or even lexical. Looking for information on the web, using search engines, is done by finding groups of terms in the pages and in the documents, without taking consideration of the meaning of those terms.

For example, using the most popular search engine, Google™, to look for information about deep wells, brings the following results on the first page *(Fig. 6)*.

The results page mixes information about 1) drinks, 2) phenomena in quantum mechanics, 3) oil well injection, and (4) electronic components. One might wish to go to the "advanced search" page and specify that only websites about oil wells should be returned. This is not possible, since Google™ doesn't allow this restriction. As a matter of fact, none of the most popular search engines currently used could restrict search to a category of pages, as the semantics of the pages are unknown to them.

Supported by the W3C of which it is a priority action, many projects aim at developing the semantic level, where information is annotated by its meaning. A necessary stage is

```
Deep Well
<http://pages.britishlibrary.net/edjason/deep/>
Find a Well Drink Deeply. Subscribe to Deep_Well. Powered by groups.yahoo.com.

Infinitely Deep Well Java Applet
<http://www.nhn.ou.edu/~mason/quantum/deepwellmain.html>
This applet illustrates the infinitely deep (square) potential well in
nonrelativistic quantum mechanics.

Deep Well Injection
<http://www.frtr.gov/matrix2/section4/4-54.html>
Deep Well Injection (GW Containment Remediation Technology).

Deep Well Replicator Starter Kit
<http://www.vp-scientific.com/96_deep_well_replicator_starter_kit.htm> -
Two Deep Well Grooved Pin Replicator. VP OK9A - Starter Kit.
```

Figure 6

The need for semantics.

to define consensual representations of the terms and objects used in the applications, these consensual representations are called ontologies. These ontologies will be expressed in OWL (Ontology Web Language) which itself is based XML and RDF (Resource Description Language, a specialisation of XML). Programs the world over support this movement towards the semantisation of information. In Europe, the EC provides strong support through the *Information Society Technologies (IST)* programme. A few ontology development projects have taken oil and gas industries as their application domain. A good definition of ontologies is provided in the "Web Ontology Language Use Cases and Requirements" document published by W3C (2004): *"…ontology defines the terms used to describe and represent an area of knowledge. Ontologies are used by people, databases, and applications that need to share domain information (…) Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable.*

The word ontology has been used to describe artefacts with different degrees of structure. These range from simple taxonomies to metadata schemes, to logical theories. The Semantic Web needs ontologies with a significant degree of structure. These need to specify descriptions for the following kinds of concepts:

- *classes (general things) in the many domains of interest;*
- *the relationships that can exist among things;*
- *the properties (or attributes) those things may have".*

The definition of ontologies is a multidisciplinary work, which requires competence:

- in the application area: geographical information, subsurface objects, facilities, transportation, processes, chemistry, environment, etc.;
- in the modelling of knowledge into a form exploitable by machines.

It is also an important stake for the actors of the field, who will use the standards defined to annotate and index their documents, their data, their codes, in order to facilitate the semantic retrieval.

Thanks to shared ontologies, future interoperability will involve the dynamic discovery and use of software services based on metadata such as supplier, price, performance, quality of service, semantically enabled input-output descriptions, and possibly on standard representations of the software's internal processes. We see already such elements in web services technologies as shown in the article by Belaud *et al.* in this issue, and in Leal's article with the use of RDF and OWL. Ongoing IT projects such as DAML-S, an ontology of services for software agents (Ankolenkar *et al.*, 2002), and SWWS, Semantic Web Enabled Web Services (Swws, 2004), are stepping stones on the path to semantic interoperability.

Challenges in Interoperability

Software interoperability poses several challenges to the profession. In order to harvest the maximum benefits from the opportunities given by mixing and matching software components from different sources, users and vendors need to work together and address a number of issues before moving into full production.

- Quality of Service (QoS): suites of interoperating software components need to provide a QoS matching those of conventional software suites. This implies not only working on the quality of individual software pieces, but also being able to assess the quality of software assemblies, which is still an open issue in the general case.
- Debugging facility: it should be easy to identify the source of a problem when things go wrong. When a software is a combination of tens or hundreds of components, tracing errors from symptoms presented to users to root causes might not be easy.
- Performance: interoperability relies on layers of middleware implying a usually moderate overload; in addition, combinations of components which have not been designed together could be less efficient than combination of native components from the same supplier. This issue can be a serious drawback in demanding applications.
- Responsibility: when acquiring software from distinct sources, a user will still expect the same level of support in case of problems. However a single vendor would not guarantee someone else's component in most cases. This is an opportunity window for systems integrators who could build interoperating software suites for customers and provide the needed support.

When solutions will be provided to the above-mentioned challenges, engineers and technicians in the oil and gas industries will enjoy the many benefits of interoperability. We hope that the current issue of *Oil & Gas Science and Technology - Revue de l'IFP* contributes to these exciting developments.

REFERENCES

Ankolenkar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R. and Sycara, K. (2002) Daml-s: Web Service Description for the Semantic Web. *First International Semantic Web Conference (ISWC)*, Sardinia, Italy.

Brandenburger, A. and Nalebuff, B. (1996) *Co-Opetition*, *Currency Doubleday*, New York.

Braunschweig, B., Pantelides, C.C., Britt, H. and Sama, S. (1999) Open Software Architectures for Process Modelling: Current Status and Future Perspectives. *FOCAPD'99 Conference*, Breckenridge, Colorado.

Braunschweig, B. (2002) The Cape-Open 1.0 standard, March 2002, www.colan.org.

Braunschweig, B., Irons, K., Köller, J., Kuckelberg, A. and Pons, M. (2001) Cape-Open (CO) Standards: Implementation and

Maintenance, *Proceedings of 2nd IEEE Conference on Standardization and Innovation in Information Technology*, Boulder, Co, 335-338.

Braunschweig, B. and Gani, R. (2002) *Software Architectures and Tools for Computer-Aided Process Engineering, Computer-Aided Chemical Engineering,* **11**, Gani, R. (ed.), Elsevier, Amsterdam.

CO-LaN (2002) *Cape-Open Laboratories Network Web Portal*, www.colan.org.

FTC (2004) *United States Federal Trade Commission Decision and Order (public record version), Docket 9310 Aspen Technology Inc.*, July 2004, www.ftc.gov.

Harter, C. (2004) *Herding Cats - the Challenge of Data and Application Integration,* First Break, **22**, 57-61.

Mandil, C. (2000) *Personnal Communication.*

Object Management Group (2004) *Uml, Unified Modeling Language.* http://www.uml.org/

Sessions, R. (2004) The Road to Interoperability. *Object Watch Newsletter*, www.objectwatch.com

Sws (2004) *Semantic Web Enabled Web Services,* http://swws.semanticweb.org

W3C (2004) *Web Ontology Language Use Cases and Requirements*, http://www.w3.org/TR/webont-req/

Final manuscript received in May 2005