Consuming CAPE-OPEN Thermodynamics from Multi-threaded applications

Dr. Jasper van Baten, AmsterCHEM, Las Rozas, Spain

Summary

Applications that perform chemical process or equipment calculations may distribute calculations over multiple nodes to benefit from modern computer hardware. In the context of desktop simulation applications, this presentation focuses on multi-threaded access to third party thermodynamics servers via CAPE-OPEN interfaces. Threading models are discussed as well as other implementation specifics that affect computational performance; multi-threaded thermodynamic access has been successfully implemented in the COFE simulation environment. This implementation, and other emerging applications, highlights the difficulties that may arise from issues with the thermodynamic server implementations that are currently available. It is expected that multi-threaded computation using thermodynamic servers will gain momentum and such issues will be addressed by the thermodynamic software vendors.

Introduction

Thermodynamic property and phase equilibrium calculations are at the root of many – if not most – chemical engineering process and equipment calculations. Examples include steady state and dynamic process flowsheet calculations, used for process design, analysis of existing processes, process control and operator training, or simulations of chemical equipment such as reactors and separators for design and retrofit purposes. Often, such models are based on first principles, where mass and energy balances must be closed and transport properties such as thermal conductivities and viscosities are required to provide details on mass and heat transport. For process simulations, typically more than 90% of the computational time is spent in thermodynamic calculations.

Some simple steady state flowsheet calculations or equipment models can converge within seconds of CPU time, but as soon as such process calculations have to be solved in the context of goal seeking or optimization CPU time drastically increases. Other types of calculations, such as dynamic flowsheet calculations, oil well reservoir calculations or Computational Fluid Dynamics (CFD) type simulations using transport properties that are based on thermodynamic sub-calculations typically require several orders of magnitude longer computation times, and development of efficient strategies of such calculations are warranted.

Trends in computing equipment

Moore's law[1] indicates that the number of transistors that are placed on common computational equipment approximately doubles every two years. As a result of that, computational speed of computers increases exponentially with time. This is illustrated in Figure 1. If we have a closer inspection of this figure, we note that in recent years, computers include multiple cores per node. This trend implies that we cannot benefit from the exponential increase in computational performance by sticking to the classic approach of a single-threaded calculation; such a calculation will only occupy a single core on a single node. Hence, to allow for computations that benefit from modern-day computing equipment, we need to somehow distribute our calculations over different nodes. Depending on the increase of computational speed on a single node or core was the main source of making computations faster up to

roughly the end of 2004[2]. Computers with nodes that contain up to 16 cores are not exceptional today.

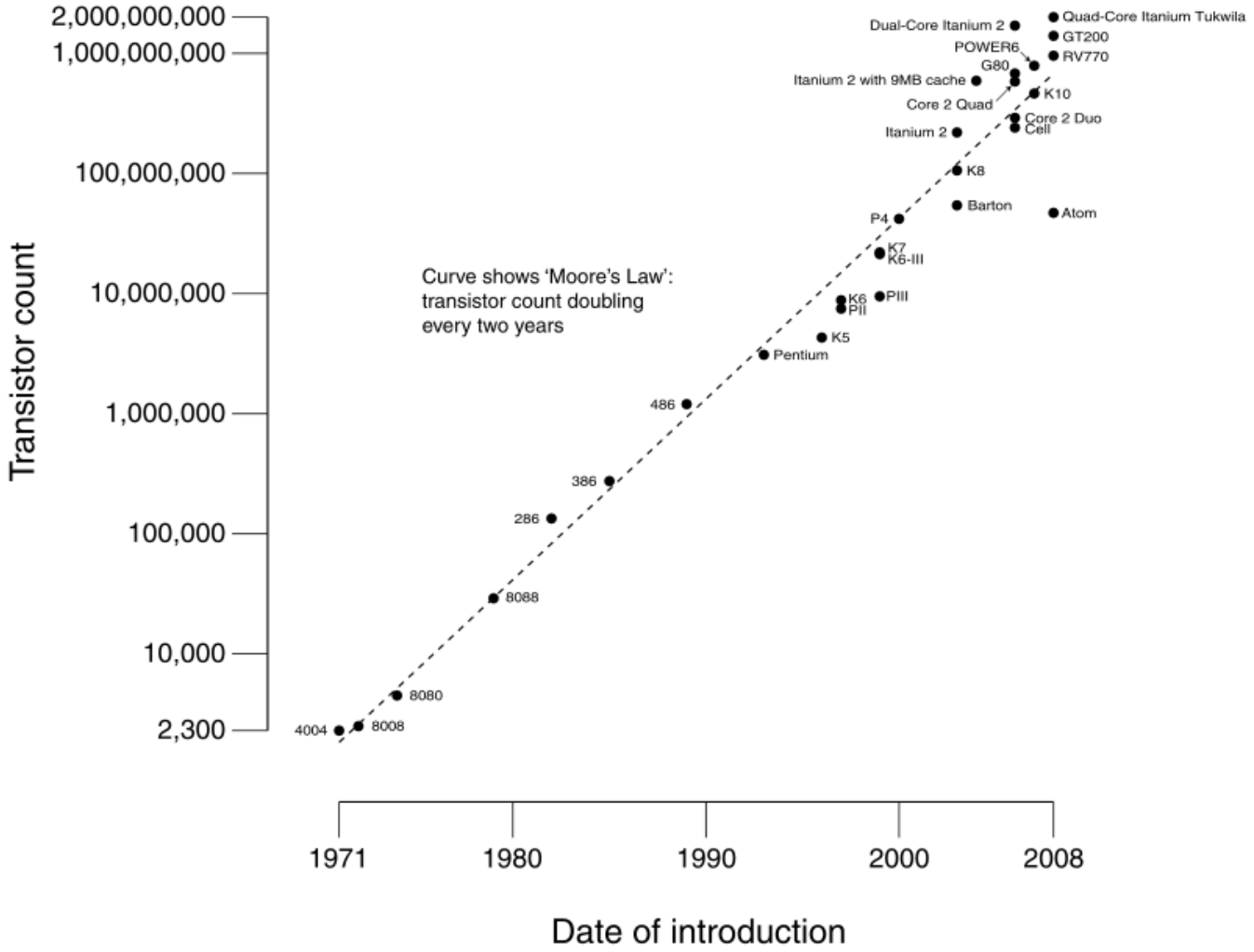

Figure 1: CPU transistor Count 1971-2008 & Moore's Law. Transistor counts for integrated circuits plotted against their dates of introduction. The curve shows Moore's law - the doubling of transistor counts every two years. Image from Wikipedia Commons, Nov 2008, by author Wgsimon.

Distributed vs. multi-threaded calculations

The above implies that we need to distribute our calculations over multiple nodes, or over multiple cores at the same node. Using multiple cores (within the same computer or distributed over multiple computers), we can distribute the process calculations over multiple processes. Using this approach, the calculations can be performed in a highly scalable manner[3]. Generally the communication between nodes is done via TCP/IP, either over the internal loopback network device, or over a LAN or even WAN. More advanced communication hierarchies are available these days, such as InfiniBand[4].

In the limit of a few nodes and independent calculations, the calculation time is split in two with each doubling of the amount of nodes used. In the limit of many nodes and dependent calculations, the data traffic between the nodes is the limiting factor, and increasing the amount of nodes eventually actually increases the calculation time. Hence, there is an optimal

number of nodes for each process. Often the amount of nodes actually used is dictated by the nodes available to the user rather than by this optimum.

The typical process engineer performs calculations using a desktop application at his own assigned computer. Typically, such a computer is a desktop computer equipped with a single node with multiple cores. In such cases, the number of nodes and cores is dictated by the equipment at hand, and to make optimal use of it, multi-threading applications are interesting. Using technologies like HyperThreading[5], the threads are automatically distributed over the available cores; if we have an appropriate number of threads that perform the calculations, the full computer capacity can be employed. As opposed to distributed calculations, multi-threaded calculations are performed by a single process. Hence, the calculations are all performed in the same process space, having access to the same process memory. The communication between the threads can therefore be very efficient. This presentation focuses on multi-threading rather than distributed calculations.

Dividing the load

Dividing the load of the computation at hand over the available threads can be a challenging task, and is highly dependent on the type of computation that needs to be performed. Generally, a good division of the load is one that most equally distributes the calculation time required, while minimizing the dependence and therefore the communication and synchronization between the threads.

The nature of the computation at hand often provides a good clue on how to distribute the load over different threads. For grid-based calculations, such as CFD for example, the division is commonly made by splitting the computational domain over the threads; the size of the boundaries between all of the sub-domains represent the required communication between the threads; the computational load of each thread is proportional to the size of each sub-domain. For calculations that are iterative and with some level of independence of the iterations (e.g. perturbation of a system to obtain its sensitivities, or repeated simulations with different inputs such as the parametric studies performed in the COFE simulator[6]), the independent iterations can be divided over different threads. No general recipes are available for distributing computational load over multiple threads.

As the division is performed with the aim of independence of calculations, equal loads and minimal communication, it is likely that each thread will have to perform its own thermodynamic calculations.

Thermodynamics and CAPE-OPEN

Traditionally, simulation applications come with built-in thermodynamic subroutines that perform the thermodynamic property and phase equilibrium calculations. Chemical companies that have experience with particular processes that they are working with may have their own in-house thermodynamics. Thermodynamic companies may specialize in particular thermodynamics that they provide commercially, and may be used by chemical companies in the process simulations. Also, chemical companies may want to perform their process simulations using the simulation software from a particular vendor, but require thermodynamics from a different vendor. Nearly all simulation applications therefore provide a means to incorporate third party thermodynamic routines.

A common way of introducing third party thermodynamic routines into simulation applications is by means of custom interfaces; the simulation user frequently has to provide a FORTRAN interface routine that takes care of performing the calculations or dispatching the calculations to the third party thermodynamic software that is used. Such interface routines are

specific to both the simulation environment and the thermodynamic software used. For a different simulation application, or when using different thermodynamics, new interface routines have to be written. With software upgrades, the interface routines may require maintenance.

CAPE-OPEN[7, 8] provides a means of avoiding these custom interfaces. CAPE-OPEN is a collection of interface standard definitions that facilitates exchange of thermodynamic and physical property models and unit operations (Process Modeling Clients, PMCs) between available simulation applications (Process Modeling Environments, or PMEs).

Currently, most main-stream steady-state simulation environments support importing third party thermodynamic servers via CAPE-OPEN. CAPE-OPEN thermodynamic interfaces have been well established, validated and demonstrated[7-17]. Examples of simulation environments supporting external CAPE-OPEN thermodynamics servers include Aspen Plus (http://www.aspentech.com/)[18], Aspen HYSYS (http://www.aspentech.com/)[18], BASF ProcessNet[19], ChemStations ChemCAD (http://www.chemstations.com/), COCO (http://www.cocosimulator.org/)[6], COMSOL Multiphysics (http://www.comsol.com/)[20], Honeywell Unisim Design (http://www.honeywell.com/ps/unisimdesign), ProSim ProSimPlus (http://www.prosim.net/)[21], PSE gPROMS (http://www.psenterprise.com/gproms/)[22], SimSci Pro/II (http://iom.invensys.com/) and SolidSim (http://www.solidsim.com/)[23].

In order to reduce overhead of software creation and maintenance by elimination of custom interfacing, it makes sense to implement a CAPE-OPEN interface for thermodynamic servers as well as for simulation applications that consume thermodynamics.

CAPE-OPEN architecture and threading models

The CAPE-OPEN interfaces are defined as a common object model structure, and there are two implementations available; COM and CORBA. COM[24] (Common Object Model) is integrated with Microsoft Windows. CORBA[25] (Common Object Request Broker Architecture) is in principle platform independent, but requires additional software to be installed to establish the connection (Object Request Broker (ORB) software). Of these two implementations, COM is currently the widely used one, and only very few CORBA CAPE-OPEN implementations are available. Thus, a COM implementation is the focus of this presentation.

The CAPE-OPEN thermodynamic standards at this point in time have two versions[26, 27]; Version 1.0 and Version 1.1; the former is still more widely supported, but Version 1.1 has a better design and solves a number of issues that were not well dealt with by the Version 1.0 specification. These interface specifications not only allow for simulation applications to interface with external thermodynamic servers, but also cater for the thermodynamic interactions between CAPE-OPEN based unit operation models embedded in CAPE-OPEN aware process simulators.

Nearly all CAPE-OPEN thermodynamic servers are available as in-process (DLL) COM objects. COM threading models are well defined, and the discussion of threading models would apply to other middle-ware as well. COM threading models come in several flavours, two of which are interesting in this context:

- *Apartment threaded*: each COM object created can only be accessed from the thread it was created from. Therefore, for each COM object it is guaranteed that it is not accessed from multiple threads simultaneously. Multiple single-threaded apartments can however exist.
- *Free threaded*: each COM object created can be accessed from any thread. Each object can therefore be accessed simultaneously by multiple threads.

The COM client (the simulation application consuming the thermodynamics) and the COM server (the thermodynamic server providing the property and phase equilibrium calculations) can each choose their threading model. The COM mechanisms implemented by Windows takes care that the communication between the server and the client is done in such as way that the threading requirements for both server and client are satisfied. Table 1 illustrates this:

Table 1: requirements for synchronization of COM calls between client and server. Yes in this table indicates that Windows synchronization mechanisms will be automatically invoked.

	Client Apartment threaded	**Client Free Threaded**
Server Apartment Threaded	No	Yes
Server Free Threaded	No	No

For a multi-threaded application, it would be intuitive to initialize the COM communication as Free Threaded. However, unless the server is implemented as Free Threaded, we see from Table 1 that synchronization is required. Synchronization implies that Windows will make sure all the calls to a COM object are performed by the thread that created the object (as expected by the Apartment Threaded model). For applications in which the thermodynamics consume a considerable amount of the total CPU time, the performance of a multi-threaded application is in this scenario drastically reduced, as each thread has to wait for other threads to complete the thermodynamic calculations. This is even worse in the case where the thread that created the COM server object is one of the calculation threads; this thread will now bear nearly the entire computational load.

Unless we can guarantee that the thermodynamic server is Free Threaded, the thermodynamic property and phase equilibrium calculations are no longer direct calls into the thermodynamic server. The software that is in between the client and the server to synchronize the calls is called a marshaller. Marshalling thermodynamic calls is in general detrimental to the performance of the calculations and should be avoided. Nearly all CAPE-OPEN COM thermodynamic servers that are currently available are implemented as Apartment Threaded. The inescapable conclusion is that the client should access the server according to the Apartment Threaded model.

As the Apartment Threaded model requires that we can only access the COM object from the thread it was created in, this implies that each calculation thread should create *its own copy* of the COM server. This makes sense also from the point of view of implementation of the server; the server has indicated by implementing the Apartment Threaded model that its objects are not thread-safe and should only be accessed by one object at a time. This implies that the COM server uses resources (local data, global data, …) that are not suitable for multiple accessing at the same time. For illustration, a property evaluation may depend on temperature, pressure and composition. While evaluating the property, the server will obtain temperature, pressure and composition from the client and store them during the evaluation. If a second evaluation would be started while the first evaluation has not yet finished, the values for temperature, pressure and composition will be overwritten by the second calculation, and the first calculation completes with incorrect data.

To conclude: the Apartment Threaded model is implemented by most servers. Unless the client can insure that the server is actually Free Threaded, it should use the Apartment Threaded model as well. Each calculation thread should then create its own COM object.

Thread safe or not?

A common assumption in thermodynamic software implementations is that the software is only accessed from a single thread. Even in the case of the Apartment Threaded model, multiple threads may exist that each create their own COM object. This is the scenario as outlined above as desirable to speed up process calculations. However, if the COM server is using global data then this will be shared by all threads and concurrent access of separate COM servers from different threads may cause invalid results or software crashes. By advertising the Apartment Threaded model, such a COM object does not do what it promises, and the thermodynamic software vendor should fix such problems.

Data marshalling

We have seen that threading models can give rise to marshalling, and marshalling should be avoided due to its adverse effect on thermodynamic calculation performance. Other factors can result marshalling: if the data that is accessed by the client does not occupy the same memory as the data accessed by the server, data marshalling is required. The COM machinery in Windows will also automatically take care of this, but it should be avoided for performance reasons. The data marshalling required for a simple property evaluation (such as density for the vapor phase at given temperature, pressure and composition) can take as long as the calculation itself.

To avoid data marshalling, the COM server should be loaded into the same process space as the client, thereby being able to access the same memory directly. The COM server is therefore best written as an in-process server (DLL). Out-of-process servers (EXE or service) are marshaled via the Windows RPC mechanism.

Another reason for data marshalling is that the *data formats* required by the server and the client are different. A common example of this is software components written in the .NET framework and running in the Common Language Runtime (CLR); such software components have built-in support for COM, but at the cost of data-marshalling (unless both the client and the server are written in .NET). As nearly all process simulators are using calculations that are native (not .NET), thermodynamic servers are best implemented outside the .NET framework as well.

Persistence

Depending on the thermodynamic server, the end-user may be offered the option to modify the thermodynamic server configuration while it is being used. For example, the end-user may add compounds to the package, or change thermodynamic model selection or model parameters. At the start of the calculation, the configuration of the thermodynamic server must be transferred to the calculation threads, in order for the thermodynamic calculations to be consistent with the user's configuration changes. CAPE-OPEN provides a mechanism for persistence of thermodynamic server configurations, and this mechanism provides the solution. The thermodynamic server calculation can be saved in the thread in which the user made the configuration changes. Each calculation thread then creates its own copy of the thermodynamic server. The configuration changes can subsequently be loaded into these copies from the data that was saved earlier.

64 vs 32 bit implementations

To complicate matters, 64-bit DLLs cannot be loaded into 32-bit simulators, and vice versa. Hence, 64-bit clients can only communicate with 32-bit servers if the server is out-of-process,

which should be avoided. Currently, most simulation applications are 32-bit. More and more 64-bit simulation applications will come out, as the demands on memory requirements increase. To be able to efficiently communicate with a thermodynamic server, the thermodynamic server must in this case be available as a 64-bit version. Currently this is not the case; nearly all thermodynamics server implementations are only available as 32-bits implementations. As these cannot be directly used by 64-bit applications, and as marshalling should be prevented, thermodynamic software vendors will have to provide both a 64-bit and 32-bit implementation in the future if they are to be used efficiently with new and existing applications.

Current status

The COFE simulation environment has implemented background calculations so that the main window remains responsive during calculations and multiple flowsheets can be solved at the same time. Although not geared to reducing CPU time, this implementation is a multi-threaded thermodynamics consuming application. Furthermore, COFE provides an option for parametric studies in which the simulation is solved multiple times with different inputs, to study the effect of inputs on outputs. Here, each simulation runs in its own thread and time is essentially reduced by the amount of threads that are concurrently running. COFE therefore provides a testing platform that thermodynamic software vendors can use to evaluate operation under multi-threaded conditions. Future work will focus on reducing solution time by employing multiple threads to solve a single flowsheet.

COMSOL Multiphysics implements both a 32-bit and 64-bit simulation application where the solution to problems is obtained using multi-threaded computations. The procedure as outlined above is implemented. This provides a means of testing for thermodynamic software vendors under multi-threaded conditions, as well as a platform to test 64-bit implementations.

Currently most thermodynamic servers are available as 32-bit in-process. With the increasing requirements on memory, it is foreseen that 64-bit versions of these servers will become available.

Preliminary testing in COFE has shown that not all thermodynamic servers are thread safe. It is expected that this too will improve in the near future.

Conclusions

CAPE-OPEN provides a means of interfacing with multiple third party thermodynamic servers via a single implementation. As this reduces software development and maintenance overhead, CAPE-OPEN has found its way into many main stream simulation environments. The threading models of COM are well-defined; even though the COM technology may be fading out, the successful COM based CAPE-OPEN use in comparison to other thermodynamic interface standards is in part based on the fact that COM allows for marshalling-free high performance interactions between client and server.

To be able to make use of increasing computational performance of modern day computers, simulation applications will have to distribute the computational load over multiple cores. One possible way of obtaining this is by multi-threading.

As most thermodynamic servers are not implemented in a Free Threaded manner, multi-threading is best accomplished by creating a separate COM object for each thread that requires thermodynamic calculations. CAPE-OPEN persistence mechanisms allow for transferring thermodynamic server configurations from one thread to another.

Problems with existing implementations may arise from 32/64 bit issues and thread safety issues of existing thermodynamic servers. It is expected that these problems will be solved by the software vendors as multi-threading and 64-bit applications gain momentum.

References

1. http://en.wikipedia.org/wiki/Moore%27s_law
2. http://en.wikipedia.org/wiki/Frequency_scaling
3. http://en.wikipedia.org/wiki/Distributed_computing
4. http://en.wikipedia.org/wiki/InfiniBand
5. http://en.wikipedia.org/wiki/Hyper-threading
6. COFE (CAPE-OPEN Flowsheet Environment) is part of the COCO simulation suite. COCO is freely available from http://www.cocosimulator.org/
7. Barrett, William M. and Yang, Jun (2005), “Development of a chemical process modeling environment based on CAPE-OPEN interface standards and the Microsoft .NET framework”. *Computers and Chemical Engineering* 30, pp. 191-201
8. The CAPE-OPEN Laboratories Network, CO-LaN, http://www.colan.org/
9. Fermeglia, M., Longo, G. and Toma, L., 2009, “Computer Aided Design for Sustainable Industrial Processes: Specific Tools and Applications”. AIChE Journal, Vol. 55, No. 4, 1065.
10. Barrett, W., Yang J. and Strunjas, S., 2007, “Final Report For Verification Of The Metal Finishing Facility Pollution Prevention Tool”. EPA report, 2007.
11. Baur, R., van Baten, J., Kooijman, H. and Drewitz, W., 2006, “Cross Platform Chemical Processes”. NPT procestechnologie, Sep.
12. Dauber, F., Span, R. and Schley, R., 2010. Influence of Thermodynamic Property Models on LNG Process Simulation". gwf International - gas solutions, 13, 70-72
13. Morales-Rodríguez R., Gani, R., d´Echelotte, S., Vacher A, and Baudouin, O., 2008, “Use of CAPE-OPEN standards in the interoperability between modelling tools (MoT) and process simulators (Simulis® Thermodynamics and ProSimPlus)”. Chemical Engineering Research and Design, 86, 823
14. Fermeglia, M., Longo, G., and Toma, L., 2008, “COWAR: A CAPE OPEN Software Module for the Evaluation of Process Sustainability”. Environmental Progress 27, No.3
15. Breil, M., Kontogeorgis, G., von Solms, N. and Stenby, E., 2007, “CAPE-OPEN: An International Standard for Process Simulations”. Chemical Engineering, Dec., 53
16. van Baten, J.M. and Szczepanski, R., 2010. “A Thermodynamic Equilibrium Reactor Model as a CAPE-OPEN Unit Operation”. Computers and Chemical Engineering, in press.
17. Pingen, H., Braunschweig, B., Merk, W., Klein, R.A., Woodman, M. and Pons, M., 2005 “Breakthrough in Process Simulations through CAPE-OPEN Interoperability Standard”. NPT procestechnologie 1 feb
18. McGough, A. And Halloran, M., 2005. “Overview of AspenTech’s CAPE-OPEN Support”. Presented on 2nd Annual U.S. CAPE-OPEN Meeting, May 25-26 Morgantown, WV
19. Lohe, B., Bessling, B., Schoenmakers, H., Scholl, S., Staatz, H., 1999, “CAPE in der Praxis — Möglichkeiten und Grenzen aus Nutzersicht”. Chemie Ingenieur Technik, 71, 323

20. von Schenck, H, Andersson, G., van Baten, J.M. and Fontes, E., 2009. CAPE-OPEN Interfaces in COMSOL Multiphysics Version 4. Presentated at AIChE meeting, Nashville.
21. Vacher, A (2007). ProsimPlus: CAPE-OPEN unit operation and thermodynamic plug and socket facilities. Presented at ICheaP-8 - Ischia Island (IT) June
22. Constantinos C.P., Williams, T.H., Keeping, B.R. and Rethinam, D., 2006. “New Cape-Open Unit Operations Socket in Equation-Oriented Process Modelling Environment”. Presented at AIChE fall meeting, San Francisco.
23. Hartge, E., Pogodda, M., Reimers, C., Schwier, D., Gruhn, G. and Werther, J., 2005. “SolidSim – A new system for the flow sheet simulation of solids processes”. 7th World Congress of Chemical Engineering, Glasgow/UK.
24. Rogerson, D. (1997). Inside COM. Redmond, Washington, Microsoft Press.
25. CORBA specifications: http://www.omg.org/technology/documents/spec_catalog.htm
26. Thermodynamics and Physical Properties Interfaces, version 1.0, available from http://www.colan.org/index-33.html
27. Thermodynamics and Physical Properties Interfaces, version 1.1, available from http://www.colan.org/index-37.html