DEVELOPMENT OF AN OPEN SOURCE CHEMICAL PROCESS SIMULATOR

Raul Cota[(1)], Marco Satyro[(2)], Craig Morris[(3)], Bill Svrcek[(1)], Brent Young[(1)]

[(1)] University of Calgary
2500 University Dr, NW. T2N 1N4, Calgary, AB, Canada

[(2)] Virtual Materials Group
657 Hawkside Mews, NW. T3G 3S1, Calgary, AB, Canada

[(3)] RedTree Development
4386 Timberline Cr. V0B 1M6, Fernie, BC, Canada

Abstract

Process simulators are a key software tool used in the fluid processing industries for day to day process calculations, for process design, for process optimization and for the debottlenecking of existing processes. Current commercial process simulators do not provide the simulator's source code; the users must rely on a closed black-box approach for the unit process models. This historical approach to software development and usage is diametrically opposite to that of open source software. The open source paradigm provides a near optimal solution for the development of robust and reliable software applications while using minimal formal resources. It is recognized that the open source approach is a feasible software development solution that overcomes some of the shortcomings of the currently available commercial process simulators. This paper presents the development of a state of the art open source process simulator; as well, it also provides sample plant simulations.

Key words

Process Simulator, Chemical, Open Source, Sim42, Hydrocarbon.

1. Introduction

The importance of chemical process simulators is well documented, as they are important tools for modeling plants, while providing opportunities for optimization and debottlenecking of existing processes (Grinthal, 1993). All commercial available process simulators follow the traditional development method, which hides the source code from the users thus relying on a closed, black-box approach.

During the last decade, open source software has grown as a near optimal solution for the development of robust and reliable software applications using minimal formal resources (Moody, 2001). This trend was popularized during the development of the Linux operating system and has since moved towards end user applications. Open source is an innovative solution for the development of a state of the art process simulator that would pool talent from a large audience while providing individual users with an in-depth knowledge of its operation and the freedom to perform any desired modifications without the need to purse special agreements with software providers. This in turn fosters the development of new applications for process simulation originally unseen by the original software developers.

The underling principle of Open Source Software is program code distribution, usually by using Internet based tools, thus permitting it to be shared by all. This allows engineers to use it, test it, and take the development further and faster by providing bug fixes, changes or proposals for enhancements to the community at-large. The success of open source compared to traditional commercial software is that its testing and development base is not constrained to employees within a company, but rather it is in a sense a community driven effort. The open and free nature of such an environment nurtures the involvement of developers and users with many different needs and services varied niche interests that would otherwise be ignored by large commercial software providers.

The design of the simulator follows good software practice in which modules are created with well defined interfaces allowing for parallel and decentralized development. Individual modules can be replaced by special custom modules or by commercially available ones. For example, the simulator has a well-defined

thermodynamic interface that allows different property packages to be “plugged” in, such as the commercially available thermodynamic and physical properties engine provided by Virtual Materials Group (1).

2. "Traditional" Software practices

In this paper, the term "traditional software practices" refers to software providers that operate in a business model in which the source code of the applications is not released to the users. This "traditional" approach results in a number of shortcomings and limitations in the final products [2].

- Development base is constrained to employees within a company
- Enhancements to the application capture only the company's interpretation of the needs of their current market
- In general, users have to wait a considerable amount of time for new releases
- Specific customizations to the product can only be made by the company that developed the software product, thus creating a dependency. This is a very important factor when dealing with highly specialized and expensive applications.

In the area of chemical process applications, the currently available commercial process simulators, tend to provide a "complete solution" that involves a great deal of specialized technologies and scientific knowledge. These technologies range from highly specific thermodynamic property packages to user friendly graphical interfaces (GUI). This complete solutions approach results in software companies that must invest resources in many areas depending only on their employees and strategic partners to incorporate new developments into their products.

Software companies that focus on a small niche of the process simulation market (thermodynamics for example) may find it difficult to grow the business because they must depend on other available commercial simulators to link to their products and this may pose either technical or economic restrictions. This issue has been addressed in recent years, resulting in open software architecture initiatives (Braunschweig and Pantelides, 1996), where commercial software companies agree to standard application interfaces. “Open Architecture” is a promising generic approach to linking applications (Edwards and Merkel, 1996) without having access to source code.

3. Open Source Software (OSS)

Development of OSS is a near optimal solution for the development of robust and reliable software applications using minimal formal resources (Hecker, 1999). The key aspect of OSS is that it releases the code to the users usually via the Internet, at no cost. This allows for the rapid deployment of, testing of, development of by accepting fixes and enhancement proposals or changes.

The term "Open Source" is not controlled or owned by any individual or corporation (Scacchi, 2002). However, the term usually refers to software that is distributed under terms that comply with the Open Source Definition (OSD) [3]. The OSD is maintained by the Open Source Initiative (OSI) [4] and it specifies a few points beyond just allowing access to the code. The OSD covers points related to redistribution of work, derived work, integrity of author's code, discrimination, etcetera. See website [3] for the full text of the OSD.

The OSI maintains a list of OSD compliant licenses and many open source projects just distribute their applications under the terms of one of those previously certified licenses (Wu and Lin, 2001). At the time of this publication the OSI web site lists nearly forty different licenses. Some popular licenses are, the General Public License (GPL), the Lesser GPL (LGPL), the Berkeley Software Distribution (BSD) and the MIT license. For a more complete list of licenses refer to website [5].

4. Advantages of OSS

The open and free nature of OSS nurtures the involvement of developers and users with many different needs and the servicing of varied niches interests that would otherwise be ignored by the large commercial software providers. OSS also benefits from rapid releases and prompt feedback because it is not constrained by time zones or work hours. Also, having access to the code results in quick bug fixes that immediately become available to the community at large [2]. An additional important aspect of OSS is that the application does not necessarily depend on the involvement of specific persons or corporations. Everyone has access to the code, hence if the current management resigns, another person or group can continue the project. It has a life of its own!

In chemical engineering, it should be noted that some developments might involve crucial technology that a company can not release to the public domain for strategic or economic reasons. Therefore,

an open source process simulator must be designed and licensed in such a way that it contemplates the addition of proprietary as well as open source software applications.

5. Organization and the Sim42 foundation

The name of the newly created simulator is Sim42 (REF). The code is resident on a public server and can be accessed via CVS [7], which is also OSS and is the most common tool used for managing OSS projects (Asklund and Bendix, 2002).

Sim42 is licensed using the BSD [8] open source license. This permits the free distribution and modification of all source code and unlike some other licenses, it does not require redistribution of modifications. In other words, a company can develop and add modifications to Sim42 for their own use without being forced to share those enhancements (i.e. a proprietary model of a unit operation).

The Sim42 Software Foundation [9] was incorporated to promote the development of this OSS Process Simulator. The foundation maintains the servers that provide public access to the project source code; discussion lists and other tools that may be deemed appropriate. It also decides what code and modifications are appropriate for inclusion in new versions and is the copyright holder and license grantor for all code included in the simulator.

6. Features of the simulator

Some of the most important and distinctive features of this only OSS process simulator are:

- The simulator is written in the Python (Brueck and Tanner, 2001) programming language, which is OSS and can be obtained at no cost via the Internet. The decision to use this programming language (Hammond and Robinson, 2000) relies on the many features offered by Python. It is object oriented, it is very high level (i.e. includes an extensive set of built-in data types and functions), has a clean syntax and it is essentially multiplatform. Another advantage of Python is that it is designed in such a way that extensions can be easily added either in Python or C++. Eric Raymond provides a thorough review of Python on his website [10].

- The building blocks of the OSS process simulator were created with well-defined interfaces thus allowing parallel and decentralized development. A clear independence from user interfaces and thermodynamic method providers is emphasized.
- The flowsheet solver can propagate partial information both backwards and forwards, which allows many complex problems to be solved without iterative calculations.
- It implements a distillation column that employs a Russell (Russell, 1983) inside/out algorithm capable of solving complex pump around, side water draws and side stripper separation configurations.
- There is no need for recycle unit operations in that estimated values are used to initialize material recycle loops.
- Balances are based on "in" and "out" ports which in turn make a "Stream" just one more unit operation. Also, connections between unit operations can be made without a stream in the middle so long as an "out" port is connected to an "in" port.
- Different thermodynamic providers can be accessed within the same flowsheet or by the same unit operation.
- Multilanguage support is provided.

7. Basic design of the simulator

The simulator on its own provides an interface to simulation services such as a flowsheet object, unit operations and a thermodynamic administrator. Figure 1 shows a very basic schematic of the process simulator's design. The dashed box encloses the basic objects provided in the current distribution of Sim42. Note, that the user can communicate with the Sim42 interface directly with Python code or through a Command Line Interface (CLI). The CLI is a powerful means for users to communicate with the simulator through interactive sessions or through scripts. Graphical user interfaces (GUI) have also been developed. VMGSim, which is a commercial process simulator developed by Virtual Materials Group [10] that communicates to Sim42 through the CLI. The current distribution of Sim42 also includes Simba, an interface to Sim42 that uses a web browser. An open source GUI written in wxPython [11] is currently under development.

Sim42, figure 1, does not contain a thermodynamic calculation server. It only has a Thermodynamic Administrator, which is intended to administer the communication between the process simulator and potential Thermodynamic Servers. Writing a high quality property package is no small undertaking (Agarwal, et al, 2001) but it is expected that a number of property packages, both proprietary and open source will be added. Currently, the Virtual Materials Group [1] provides a free version of their RK property package. Figure 2 presents a schematic of the possibilities for "interfacing" to Sim42. A more detailed explanation on the basic structure of Sim42 is provided in the manual [12]

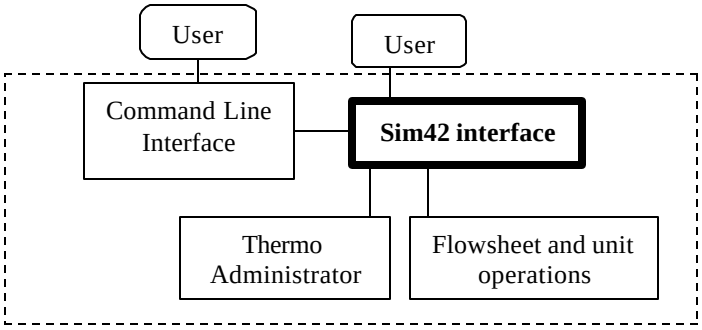

Fig1. Basic objects of Sim42

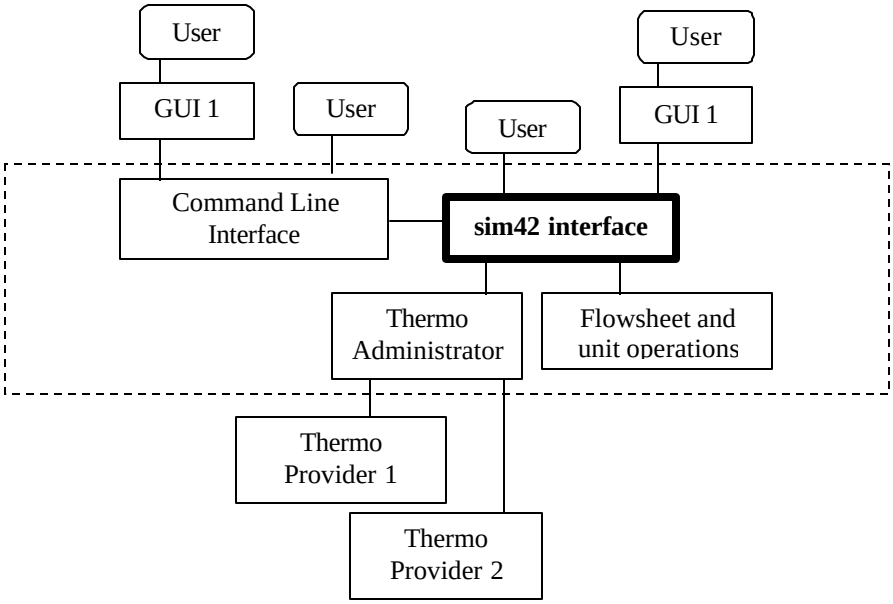

Figure 2: Basic objects of Sim42 with custom interfaces and thermo providers

8. Exploring the basic objects of Sim42

8.1. Unit Operations (UO)

Unit operations contain:

a) Child UOs; this allows the creation of complex UOs by using and connecting a number of simpler UOs.

b) Ports; These objects contain the property values and provide a unique means for the exchange of information between connected UOs. There are material, energy and signal ports. All these ports can be "In" or "Out" ports.

c) Parameters; In general, these are structural values that do not change during the solution of a simulation. Examples of such values are the number of UO inlet ports or the number of liquid phases calculated by the Thermo server.

d) Thermo Case; This object specifies the basic information needed to communicate with the Thermo Administrator, which in turn calls the appropriate Thermo Provider.

A "flowsheet" is a special UO that contains the algorithm for solving its child UOs. This algorithm recognizes the simulation structure, controls information propagation and recognizes the estimates provided for material loop recycle solution. Any UO can have any UO as a child, but it is important to note that a flowsheet must be on the top. Figure 3 presents a generic scheme of contained unit operations.

8.2. Ports

Unit operations exchange information with other unit operations by means of ports. A port is essentially an attachment point for the flow of information into and out of the unit operation. It might be a material port, which contains all of the information normally associated with a process stream (temperature, pressure, flow, composition, etc.) or it might be an energy port that just contains an energy flow or even a Signal Port that transmits a single piece of information such as a pressure drop.

A Signal Port contains a list of "Property" objects that contain such information as the name, value, conversion factor, min, max and status. "Status" indicates if the value was calculated, specified by the user, passed through a connection, etc.

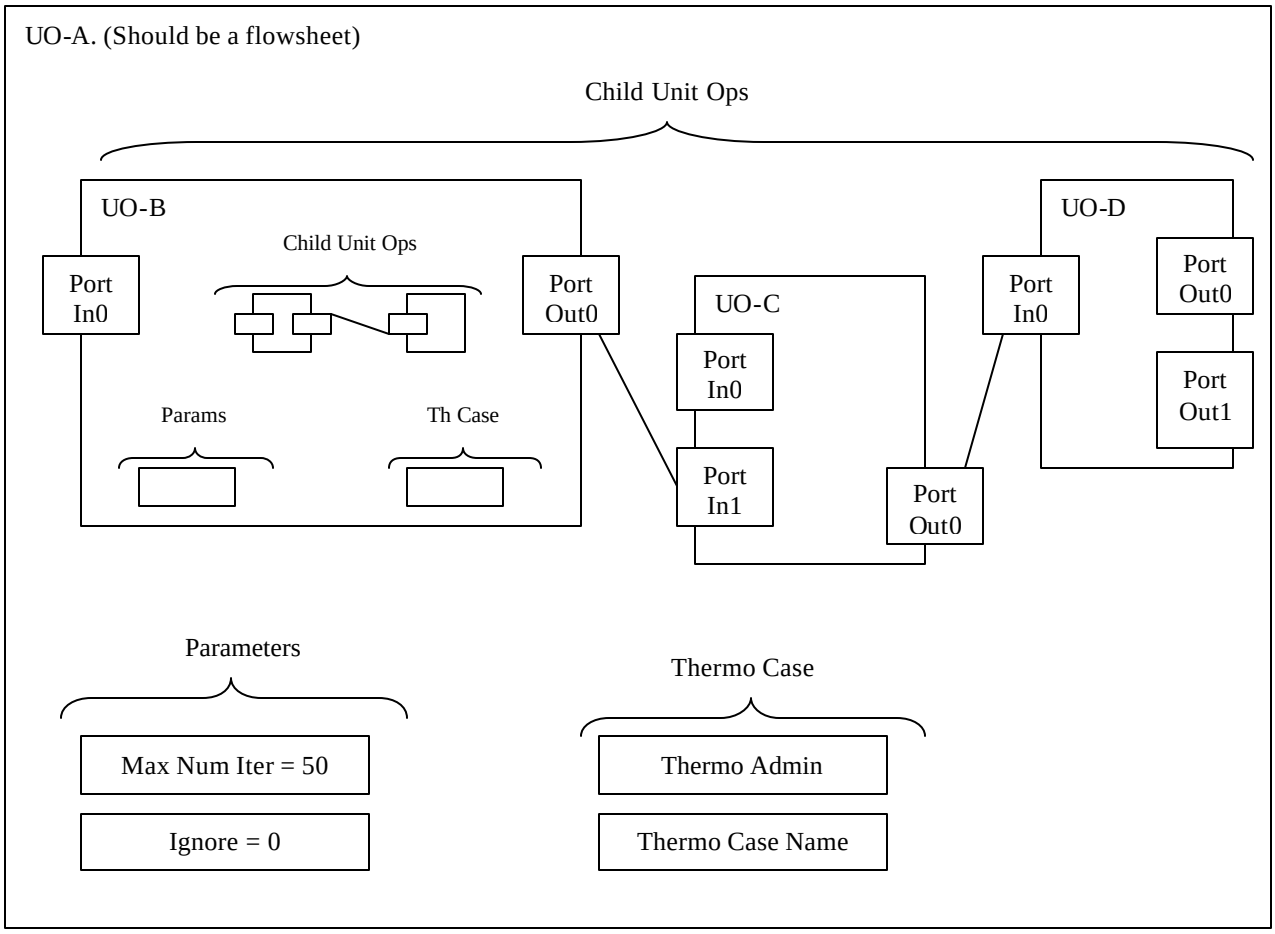

Figure 3; A generic scheme of unit operations with a flowsheet being the parent object

9. Benzene/Toluene flash calculation.

This example creates a stream and performs a flash calculation for a mixture of benzene and toluene at 80 °C and 1 atm. The simulations shown will use the interface called Simba (Sim42 Browser Assisted) which is included in the current distribution and can be run using most web browsers. Figure 4 shows a snapshot of the Simba interface. There is a multi user on-line version in the Sim42 web site which can be accessed via the following link:

WebSite: http://demo.sim42.org/sim42
Login name: guest
Password: sim42

Simba is a wrapper for the Command Line Interface (CLI) and processes commands entered into a the command text box located in the top section, Figure 4. A process simulation always has a "root" object (a Flowsheet unit operation) which in turn contains objects such as parameters or child unit operations. The "root" object is identified by the "/" character. The complete hierarchy of objects can be explored in the left column of Simba (Figure 4, simulation object tree) by clicking on the desired object. The main section of the application displays the "active" object and its deplay depends of the type of object being modified.

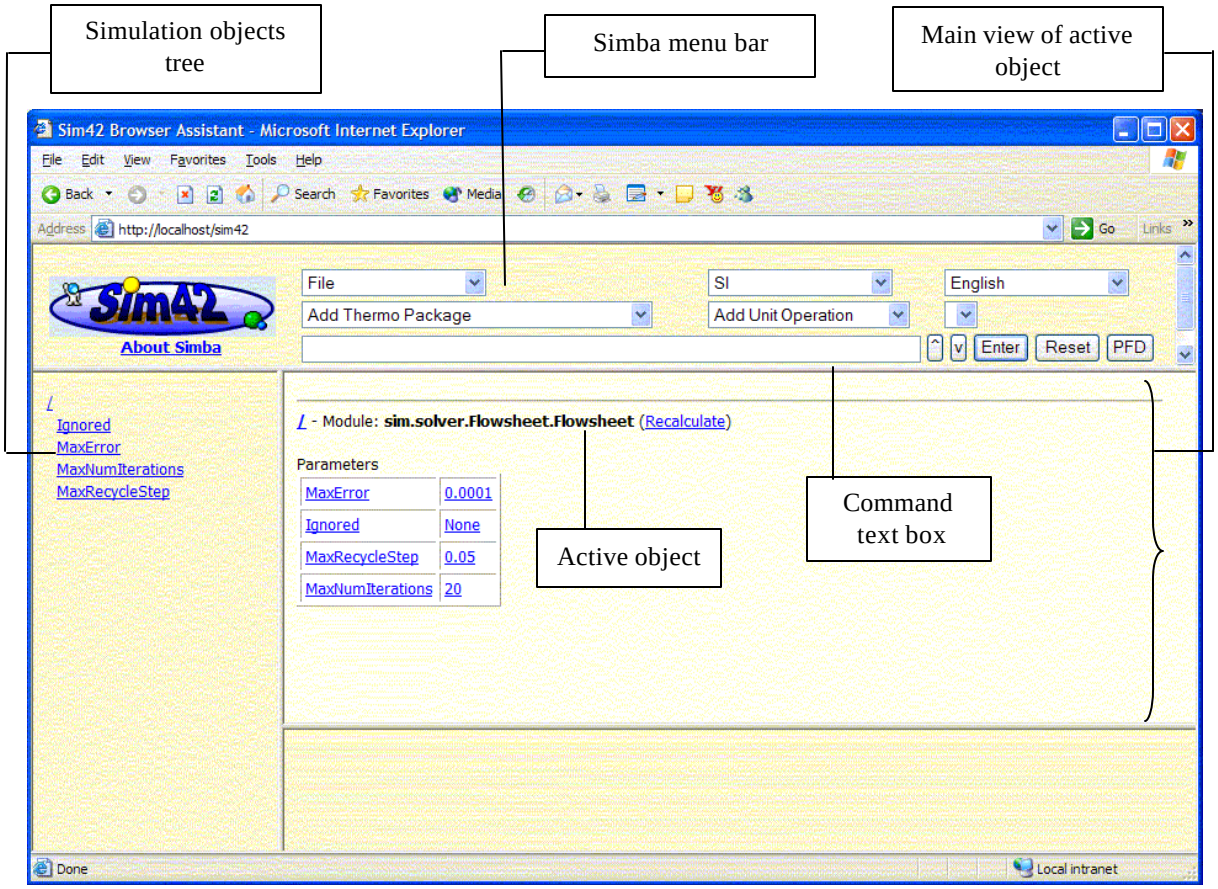

Figure 4; Snapshot of SIMBA

The first step in creating a simulation is the addition of a thermodynamic case, which in the simplest case involves the selection of a thermodynamic package and a set of chemical compounds. This can be achieved by selecting a property package from the combo box called "Add Thermo Package". In this case "VirtualMaterials.SRK" is selected and the name "ThermoCase" is assigned to it when prompted to do so. Note, the CLI command triggered for this action is:

ThermoCase = VirtualMaterials.SRK

Typing the previous line is equivalent to using the “user friendly” combo box. The following command can be typed in order to add chemical compounds to the ThermoCase (instead of selecting from the provided list). Figure 5 shows a snapshot of the added thermodynamic case.

/ThermoCase + Benzene Toluene

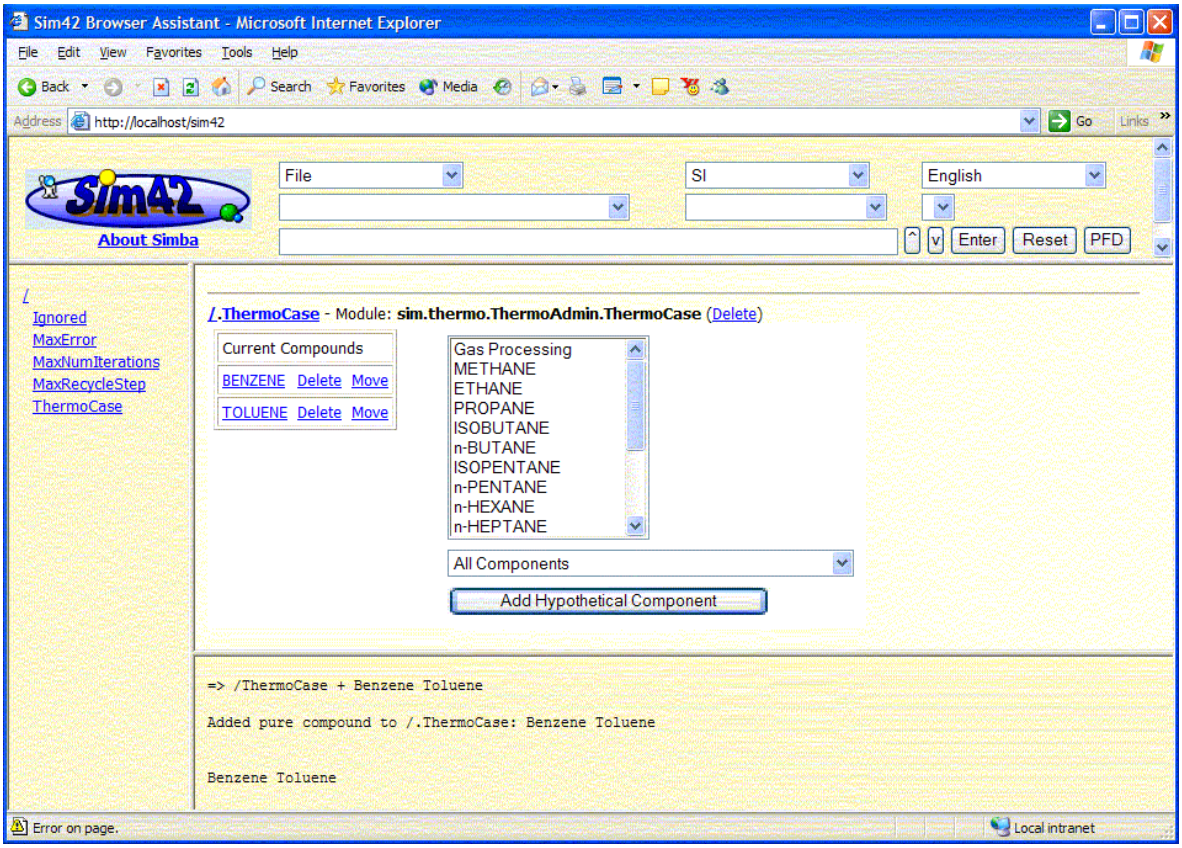

Figure 5; Snapshot of ThermoCase in Simba.

The next step in the simulation process is the addition of unit operations to the flowsheet. To accomplish this it is necessary to make the flowsheet the “active” object by clicking in the “/” located in the left column of Simba. Unit operations can be added by using the combo box labeled “Add Unit Operation”. In this case “Material Stream” is selected and the name “s” is assigned to it. The new unit operation automatically becomes the “active” object and properties can be added to its “In” or “Out” port. The displayed units use the default set “SI” but can be changed to any available set via the corresponding combo box. For purposes of this example values are input into the “In” port (P = 100 (kPa); T = 80 (C) and Composition = 0.5 0.5 (molar)). It is important to note, that all the intensive variables of the ports for the

stream appear “filled in”. Sim42 constantly monitors the degrees of freedom and performs all calculations as information becomes available. Specifying an extensive variable such as mass flow would result in a fully calculate stream. An input of 100 kg/h is used in this example and the simulation results are displayed in Figure 6.

All the previous steps could have been coded in a text file as a series of commands and then run as a script using the "Read Script" option of the “File” combo box in Simba. This same combo box provides options for storing and recalling process simulation projects, among other operations. Simba also provides an option for changing the active language to any of the currently supported languages; Spanish, French, Portuguese, Malay or English.

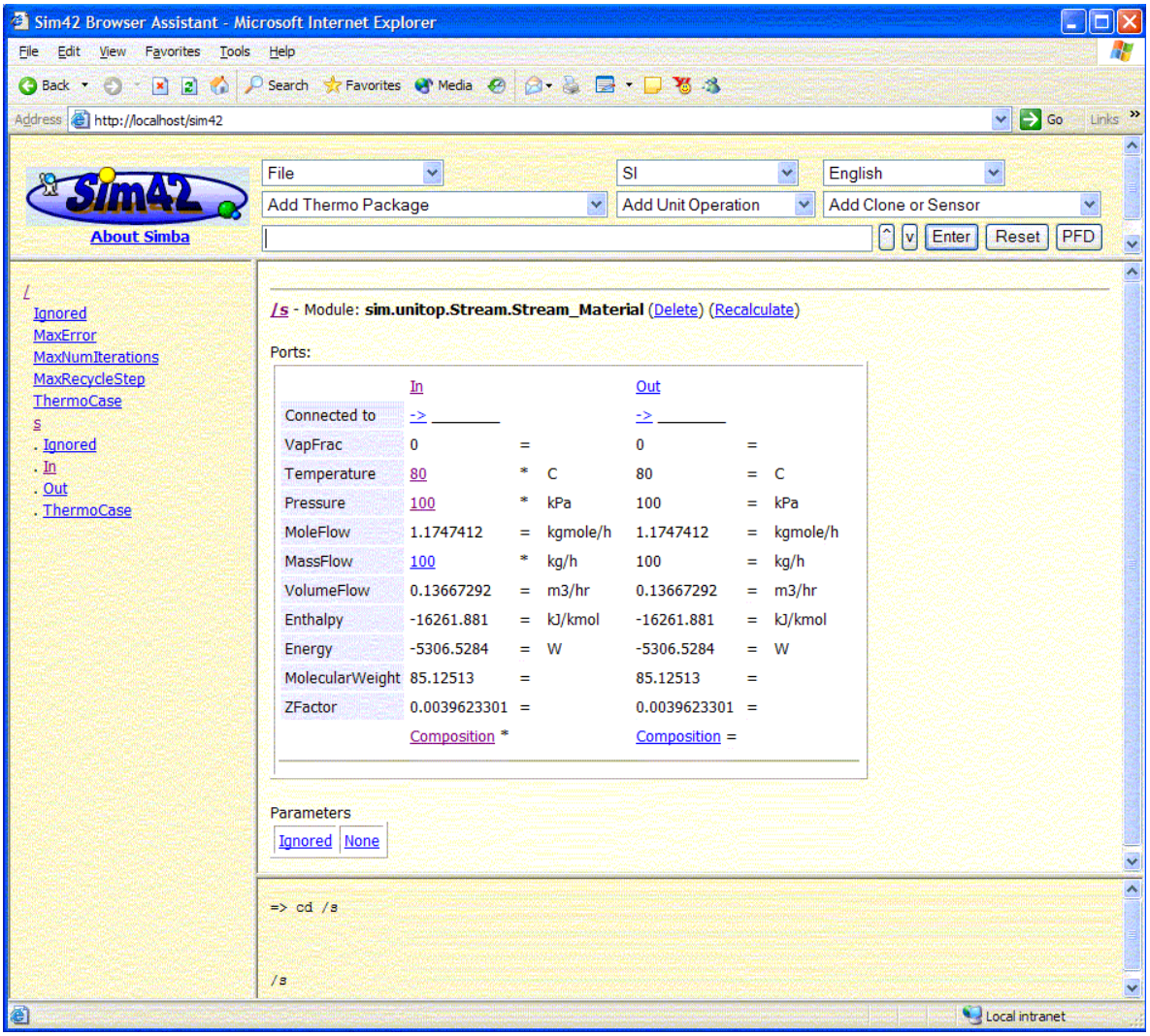

Figure 6; Snapshot of stream “s”.

10. Ethanol distillation column simulation.

This example uses live steam to fractionate an ethanol-water mixture into a near azeotrope ethanol/water at the top and a negligible concentration of ethanol in the bottoms product stream.

Again, the first step is to create the thermodynamic case, a “Feed” stream and a “Steam” stream. The values used are summarized as follows:

thermo = VirtualMaterials.PSRK
thermo + Ethanol Water

Feed = Stream.Stream_Material()
Feed.In.MoleFlow = 34.43
Feed.In.Fraction = 0.3 0.7
Feed.In.VapFrac = 0.0
Feed.In.P = 101.325

Steam = Stream.Stream_Material()
Steam.In.P = 24.7 psia
Steam.In.Fraction = 0 1
Steam.In.MoleFlow = 51.1
Steam.In.VapFrac = 1.0

In the next step a Tower is created and the name “dist” is assigned to it. The initial display of the tower is shown in figure 7. For this example, the parameter “MaxOuterLoops” is changed to 40 and twelve stages are added to the first stage. In order to add stages it is necessary to select the stage after which stages will be added. In this case “Stage_0” can be selected from the tree in the left column or the link “0” can be selected in the main view. Stages can be added using the “Add the following number of stages below this stage” text box. In this case “12” is the input.

The next step is to add a liquid draw to “Stage_0”. To do this, it is necessary to make this stage the “active” object and then select “Liquid Draw” from the “Add Feed or Draw” combo box. For this example, the name “l” is assigned to it. Adding the liquid draw makes this object the “active” object and the following values are introduced (the text written before the “>” is to indicate the active object:

/dist.Stage_0 l.Port > P = 101.325
/dist.Stage_0 l.Port > MoleFlow = 12.91

An energy draw and a tower estimate in “Stage_0” are added following the same procedure. That is: selecting “Energy Draw” from the “Add Feed or Draw” combo box and selecting “Temperature Est.” from

the “Add Spec or Est” combo box. The names “cond” and “estT”, respectively are used in this example. The value for the temperature estimate is:

/dist.Stage_0> estT = 78

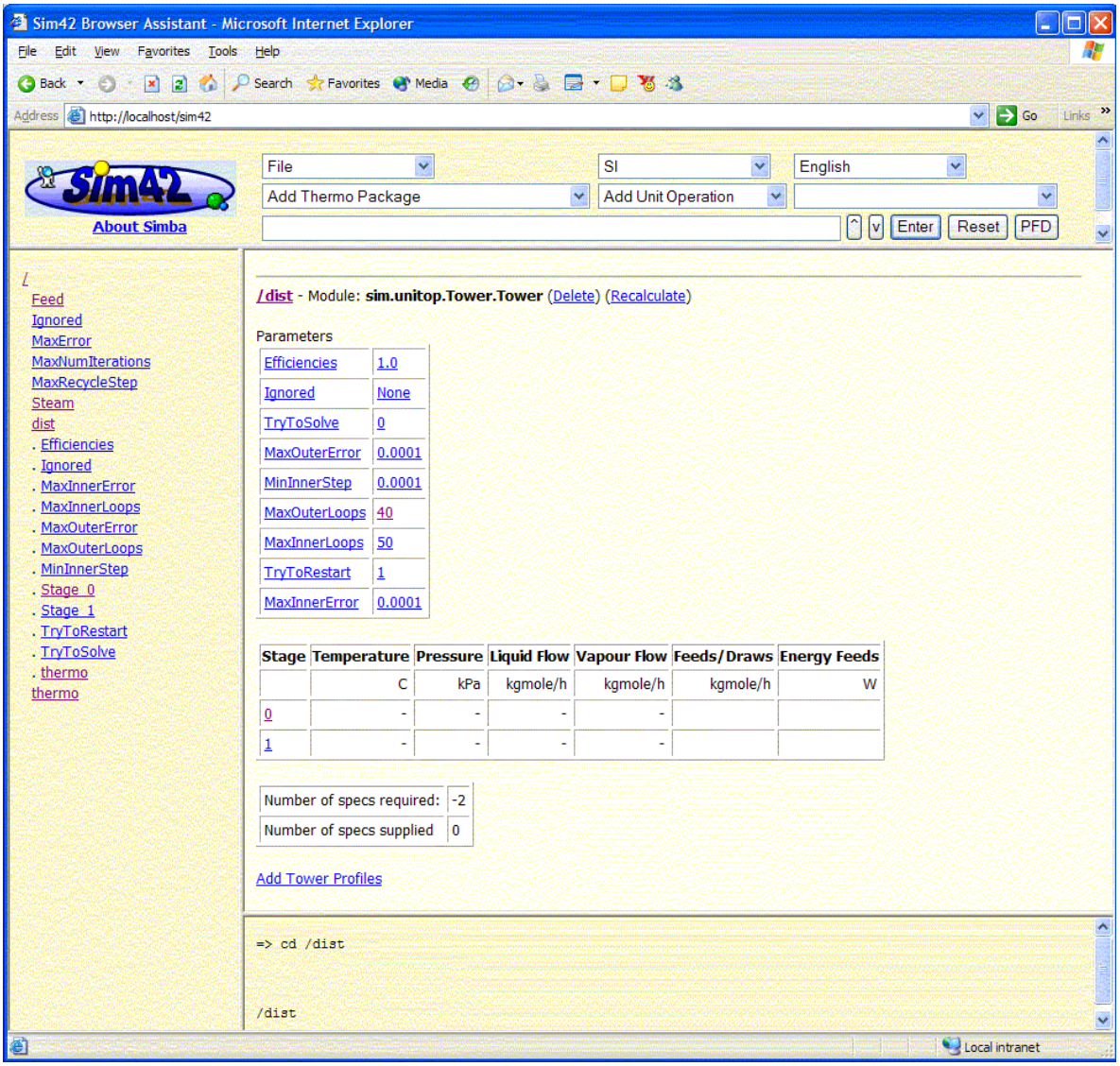

Figure 7; Initial view of a distillation column in Simba.

The next step is to add a feed to “Stage_11” of the tower. In order to do this it is first necessary to make “Stage_11” the “active” object. This can be done by selecting “dist” from the left column of Simba and then selecting “11” from the list of displayed stages. The feed is added by selecting “Add Feed” from the “Add Feed or Draw” combo box. The name “f” is used in this example. Once “f” is created, it can be

connected to the "Out" port of the "Feed" stream by clicking in the "->" link of the "Connected to" row (assuming /dist.Stage_11.f is the "active" object). It is important to mention at this point that Sim42 has a novel port-wise design in which information to unit operations is passed by using ports. In this example, the information in the "In" port of the "Feed" stream could had been input directly into the "f" port of the "Stage_11" of the Tower without the need to create an external stream.

The only ports missing from the column are the steam inlet and the bottoms product outlet. Both ports are created at the bottom of the Tower, "Stage_13". The steam inlet port is created by adding a feed and the bottoms port is created by adding a liquid draw. Both objects are added after making "Stage_13" the "active" object and are labelled "f" and "l", respectively. The newly created feed is connected to the "Out" port of the "Steam" stream. The pressure of the bottom product liquid draw is set as follows:

/dist.Stage_13 l.Port >P = 101.325

Setting temperature estimates is recommended for towers in Sim42. In this example, an extra Temperature estimate is added to "Stage_13" and a value of 100 is assigned to it.

/dist.Stage_13> estT = Tower.Estimate('T')
/dist.Stage_13> estT = 100

At this point the Tower is ready to be solved and the only thing that is left to do is setting the parameter TryToSolve = 1. Figure 8 shows the tower before solving and figure 9 shows the tower after being solved. The results of the simulation are summarized in Table 1.

Property	Feed	Steam	Distillate	Bottoms
VapFrac	0.0	1.0	0.0	0.0
T (C)	82.30	115.12	78.35	99.788
P (kPa)	101.325	170.30	101.325	101.325
MoleFlow (kgmole/h)	34.3	51.1	12.91	72.62
MassFlow (kg/h)	910.026	920.56	521.75	1308.84
H (kJ/kmol)	-26999.078	12894.15	-23523.779	-28474.91
Energy (W)	-258216.18	183025.39	-84358.88	-574401.94
MolecularWeight	26.43	18.01	40.41	18.023
Zfactor	0.00134	0.987	0.002188	0.000832
ETHANOL	0.3	0.0	0.798	0.000289
WATER	0.7	1.0	0.201	0.9997

Table 1; Results of the Ethanol production tower simulation

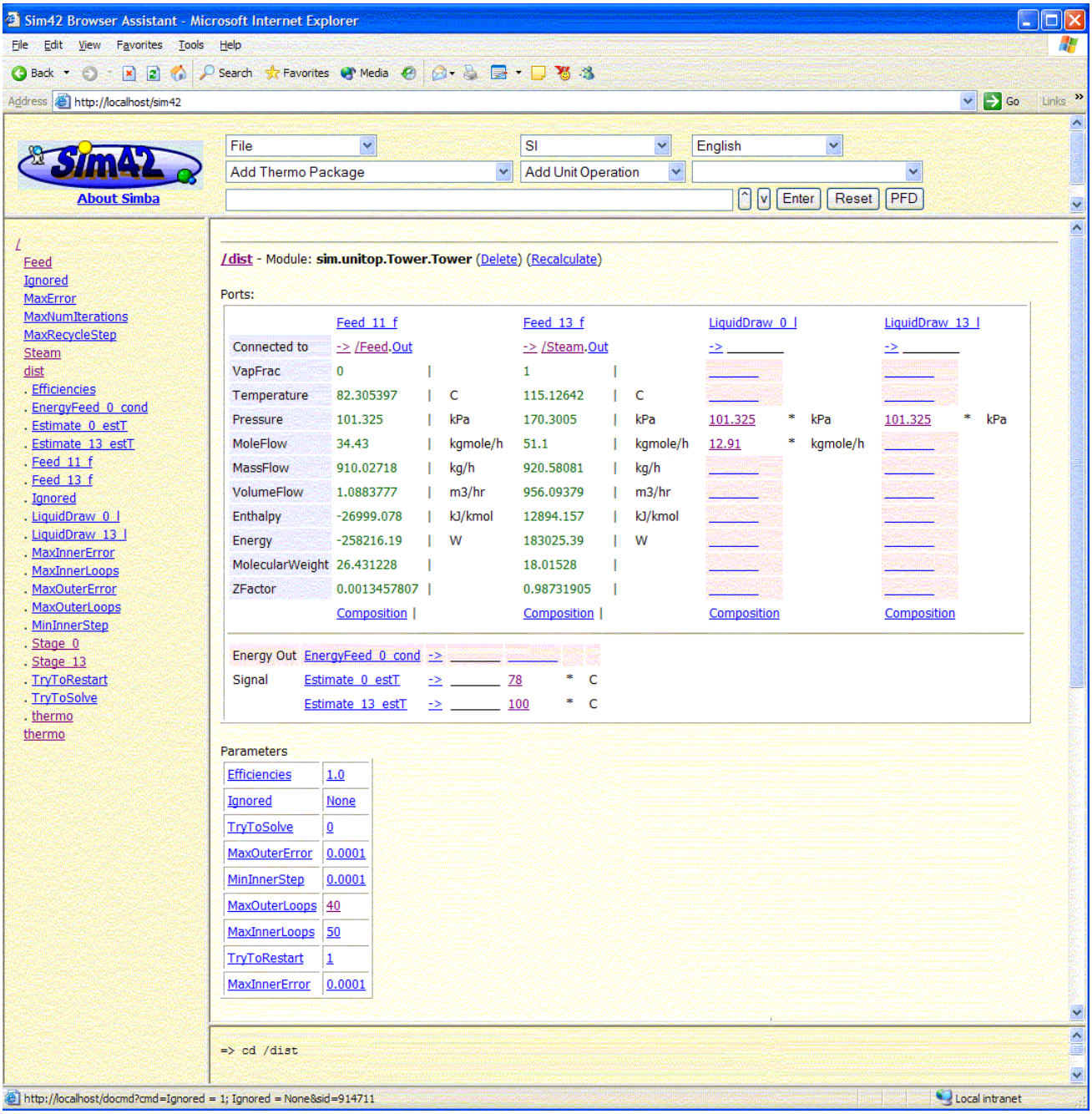

Figure 8; Simba Tower snapshot before solving.

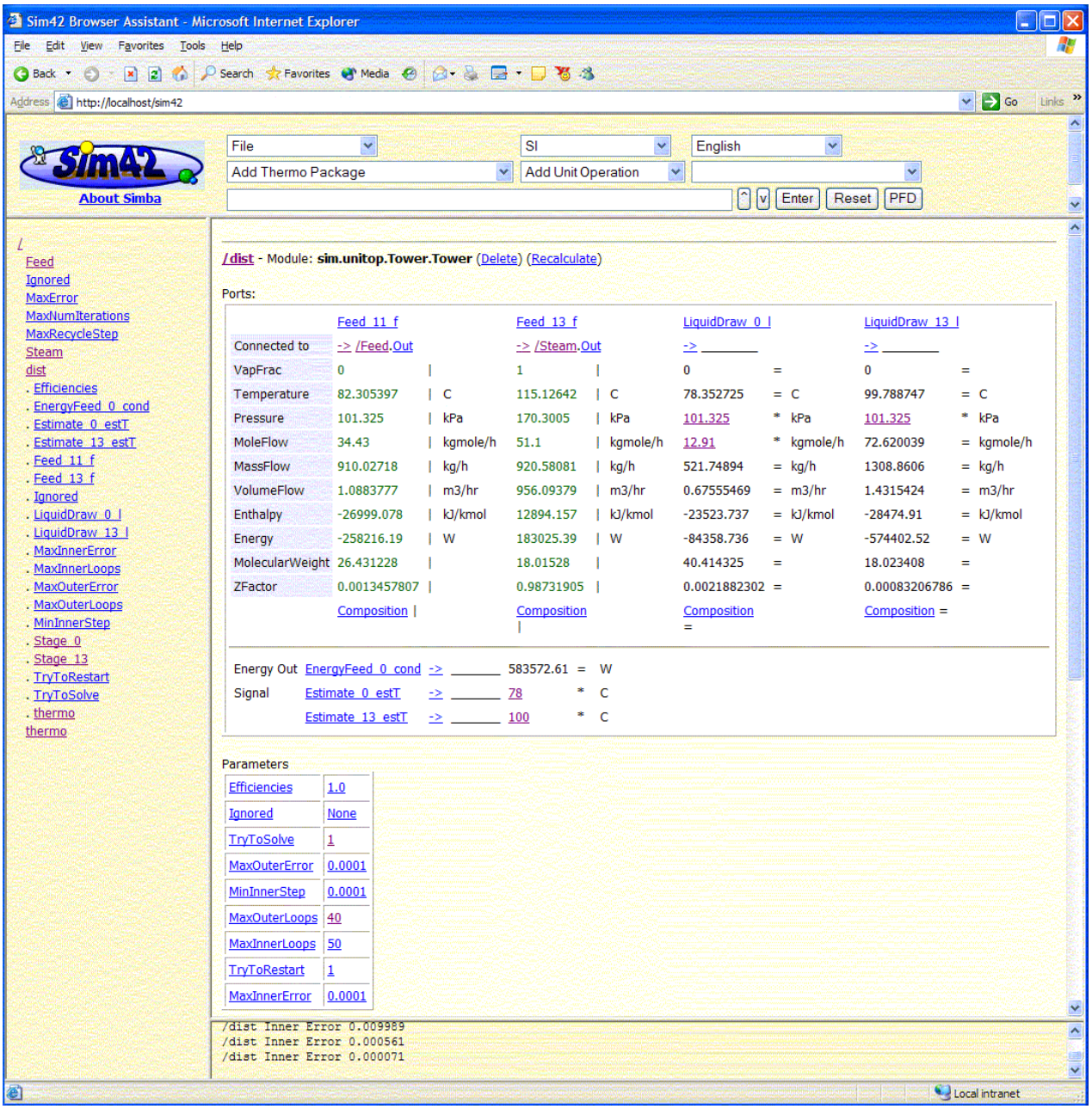

Figure 9; Simba Tower snapshot after solving.

The tower simulation is presented here to show the usage and creation of distillation towers in Sim42. It is important to note that Sim42 offers pre-built towers such as “Distillation Column”, “Rebolied Absorber”, “Refluxed Absorber” and “Absorber” which already contain ports for feeds and material and energy draws. Note, the pre-built towers can be modified like the tower described in the example ethanol/water distillation column simulation. Simba also provides a graphical representation of the

simulations which can be accessed by clicking the “PFD” button. Figure 10 shows the Simba PFD for the ethanol/water distillation column simulation.

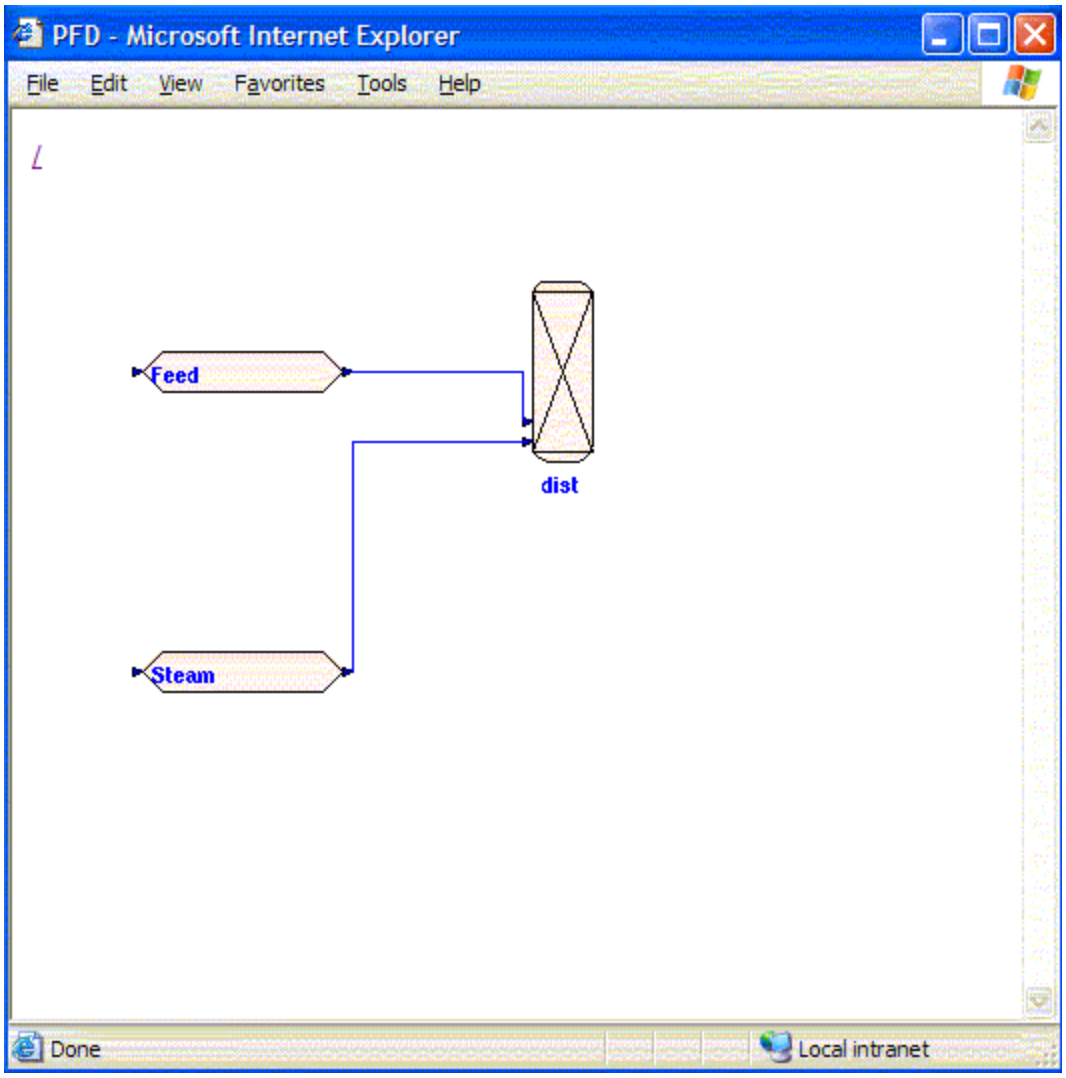

Figure 10; Simba PFD.

11. CONCLUSION

An Open Source Chemical Process Simulator was developed. It is believed that the OSS way of developing software does overcome the shortcomings posed by currently available commercial process simulators, in that Sim42 provides an inexpensive state of the art software tool for process modeling and its

object oriented design and licensing allow for easy incorporation of specific developments either open source or proprietary.

The source code for the Sim42 process simulator is available at no cost and can be downloaded from the Internet. A non-profit organization called "Sim42 Software Foundation" was created to manage the project. The development base is still small and it is expected that Sim42 will naturally grow as involvement in the project grows.

The simulator, which includes a powerful rigorous distillation tower, has been used to model a variety of plants. The examples presented here were run using Simba but other interfaces are available or are being developed including a graphical interface based on wxPython and a professional commercial interface. This flexibility is only possible because the simulator core has been designed to be independent of both user interfaces and thermodynamic property providers.

WEB SITES CITED

[1] Virtual Materials Group (VMG), http://www.virtualmaterials.com

[2] E. Raymond, The Cathedral & the Bazaar, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/

[3] Full text of the Open Source Definition (OSD), http://www.opensource.org/docs/definition_plain.php

[4] Open Source Initiative (OSI), http://opensource.org

[5] OSI Licensing, http://opensource.org/licenses/

[6] Red Tree Development Inc, http://redtree.com/

[7] Concurrent Versions System (CVS), http://www.cvshome.org/

[8] BSD License, http://opensource.org/licenses/bsd-license.php

[9] The Sim42 Foundation, http://sim42.org/bylaws

[10] E. S. Raymond. Why Python?, Linux Journal (73), 2000,

http://www.linuxjournal.com/article.php?sid=3882

[11] wxPython http://www.wxpython.org

[12] Sim42 manual, http://manual.sim42.org/

REFERENCES

- Agarwal, R., Li, Y., Santollani, O., Satyro, M. & Vieler, A., Uncovering the realities of simulation, *Chemical Engineering Progress, v97*(n5), 2001, 42-52
- Asklund, U. & Bendix, L., A study of configuration management in open source software projects, *IEE Proceedings: Software*, 2002, 40-46
- Braunschweig, B. L., Pantelides, C., Britt, H. I. & Sama, S., Open software architectures for process modeling: current status and future perspectives, *5th International conference on foundations of computer-aided process design, v100*, 1996, 220-235
- Brueck, D. & Tanner, S., Python 2.1 bible (New York, NY: Hungry Minds, 2001)
- Edwards, P. D. & Merkel, G., Impact of an open architecture environment on the design of software components for process modeling, *International conference on intelligent systems in process engineering, v92*, 1996, 307-310
- Grinthal, W., Chemputers. Process simulators shift into high gear, *Chemical Engineering (New York), v100*(n7), 1993, 153-154, 156
- Hammond, M. & Robinson, A., *Python programming in win32* (Sebastopol, CA: O'Reilly, 2000)
- Hecker, F., Setting up shop: the business of open-source software, *IEEE Software, v16*(n1), 1999, 45-51
- Moody, G., *Rebel code. inside linux and the open source revolution* (Cambridge, Massachusetts: Perseus Publishing, 2001)
- Russell, R., A flexible and reliable method solves single-tower and crude-distillation-column problems, *Chemical Engineering , v90(*n21*)*, 1983, 53-59
- Scacchi, W., Understanding the requirements for developing open source software systems, *IEE Proceedings: Software*, 2002, 24-39
- Wu, M. & Lin, Y., Open source software development: an overview, *Computers, v34*(n6), 2001, 33-38