Stephan Petersen[a], Klaus Hack[a], Peter Monheim[b], Ulrich Pickartz[c]
[a]GTT-Technologies, Herzogenrath, Germany
[b]SMS Demag, VES Melting Units/Metallurgy, Düsseldorf, Germany
[c]Billing & Loyalty Systems GmbH, Product Development, Oberhausen, Germany

SimuSage – the component library for rapid process modeling and its applications

Dedicated to Dr. Gunnar Eriksson on the occasion of his 65th birthday

SimuSage is an innovative software tool for process simulation and flowsheeting tasks. Based on ChemApp and its rigorous Gibbs energy minimizing technique, it provides a library of components for the development of highly customized process simulation models. The SimuSage concept is described, and a number of examples from typical application areas such as metallurgy, combustion technology, and other industrial high-temperature processes involving inorganic chemistry are introduced.

Keywords: Thermochemistry; Gibbs energy minimization; Process simulation; Flowsheeting; Computer aided process engineering

1. Introduction

In modeling high-temperature processes which involve chemical systems for which a consistent thermochemical description is available, one often tries to start with the equilibrium approach. Using sophisticated software tools like FactSage [1], which combine a large number of thermochemical data with powerful calculation and reporting tools, and assuming that due to the high reaction rates at elevated temperatures the deviations from the equilibrium state are small upon first approximation, such calculations are quickly set up. But in many cases real processes represent a complex combination of several subprocesses, their relationship can be modeled by flowsheets. SimuSage is a tool to model complete flowsheets using the same rigorous thermodynamic calculational methods found in FactSage.

Furthermore, processes frequently do not achieve complete thermodynamic equilibrium. Rigorous kinetic modeling in such cases has been carried out regarding some aspects, but a complete representation is nearly impossible due to the complexity of the task and the absence of reliable kinetic data. Flowsheeting with SimuSage makes it possible to create process models with an empirical component characterized by a limited number of easily determined parameters, while still complying fully with the thermodynamic restrictions.

The lack of programming tools that were deemed suitable to model complex processes in this fashion prompted SMS Demag AG, in co-operation with the former Mannesmann Datenverarbeitung GmbH and GTT-Technologies to jointly develop a suitable software package. Even at a fairly early stage the Institut für Verfahrenstechnik at RWTH Aachen University (IVT) successfully tested it in a number of complex process simulation projects and developed interesting new modeling concepts based on the new package. This process simulation tool has subsequently been made commercially available by GTT-Technologies as the software package SimuSage.

2. Requirements and specifications

Although a large variety of software tools had already been available at the time development of SimuSage started, no single tool matched more than a few of the requirements as determined by the developers. The catalog of requirements for such a new software package included points listed below.

2.1. Data availability

As mentioned above, one of the most crucial requirements was that the entire range of state-of-the-art thermochemical solution data had to be supported, without the need to simplify or reduce the complexity and quality of the data. Particularly in the area of high-temperature, inorganic chemistry, all available data should be compatible with the new tool. Fulfilling this requirement alone would mean a unique feature for the newly developed process simulation software.

2.2. Unlimited complexity of models

Essentially no limit should be imposed on the complexity of the simulations that could be modeled. In particular it should allow for iterative, nested and counter-current processes, steady-state as well as dynamic simulations, and support all types of flow control. This requires a type of software technology used as a basis that not only makes the design of complex simulations possible, but which also does not put any unnecessary penalty on the execution time of the resulting simulation.

2.3. Rapid application development

The assembly of simple processes should be as easy as possible, ideally requiring no programming at all, but only interaction with a graphical user interface. At the same time, while the user experiences increased demands as to the complexity of his or her simulations, the learning curve should not be too steep to prevent a step-by-step learning process of acquiring the necessary skills. This requirement was considered to be crucial in order for the new software to gain acceptance with engineers and scientists who are not software developers by education, but who would use it infrequently on a project-by-project basis instead of using it as the main tool in their profession. Translated into software engineering terms, this requirement would mean that a *Rapid Application Development* (RAD) environment would be necessary, which minimizes the amount of manually created code, and maximizes the use of existing components and re-use of created components.

Even a flowsheeting simulation in its early stages should have a user interface that makes it easy to demonstrate and thus increases its acceptance with users not involved in the development. In terms of the development times, it was expected that the RAD concept would need to make it possible to measure the development time of the first stages of a flowsheeting simulation in days rather than weeks. This would also make the tool essentially different from ChemApp, which at that time had already been used to model a multitude of processes. But as a pure programmer's library, ChemApp always requires the writing of code, and does not explicitly support RAD concepts, nor does it provide user interface elements to the application program.

2.4. Advanced programming techniques

At the same time, the program should be based on modern software design concepts and thus satisfy the needs of users who have a solid background in software engineering. This more advanced group of users should also be able to extend the existing software in a modular way, using their own expertise in modeling and simulation to create additional reusable components that integrate natively into the base system as supplied by the manufacturer.

2.5. Meeting the needs of the primary target group

The primary target group of this new software tool would be engineers and scientists who often use it infrequently on a project by project basis. SimuSage originated from the need to efficiently develop models for processes for which established methods of calculation do not exist – such as they do for example for distillation columns and the various types of heat exchangers in the flowsheeting of refineries. SimuSage thus concentrates on providing basic unit operations from which other, more complex process units can be built, which in turn can serve as building blocks for larger process models. A positive effect of this philosophy is that the user does not have to pay for the creation and maintenance of software tools he is never likely to use in his particular field of application.

2.6. Maximum support for user-defined code

The software should be as open as possible with respect to adding user-defined code, irrespective of whether the code is related to the simulation model, the communication of the flowsheet code with other software, or the development of a graphical user interface that is completely customizable to the users' needs. This should in particular not be impeded by an inflexible interface between the program and the user's code, which limits the user in terms of the type and complexity of custom code, or cause problems with respect to the stability and error tolerance of the resulting simulation.

2.7. Stand-alone executables of models for use by third parties

Closely linked to the previous one is the requirement that the software should meet the needs of process simulation consultants. This not only means that the user should be able to customize a flowsheeting simulation with respect to the core simulation model as extensively as possible. This requirement also implies that the user interface of the resulting process simulation should be completely customizable with the end users of the simulation in mind. Different types of end users have different needs and expectations in terms of the usability of software, ranging from a maximum ease of use, the shortest possible learning curve and the best possible protection against incorrect user input on one end, and maximum flexibility and adjustability of process parameters on the other end. The new process modeling system should allow for the creation of a flowsheeting simulation for both types of users, ideally from the same code base. Another technical requirement for consultants is that the resulting flowsheeting model can be packaged in such a way that it can be passed on as a product to the end user. An important aspect in this connection is that consultants need to be able to choose how much of their potentially proprietary simulation code is revealed or protected in the final application.

3. The SimuSage process modeling concept

In reviewing the above requirements it became not only clear that no readily available tool was a sufficiently good match; the combination of requirements also implied that it was unlikely that a monolithic, interactive process simulation program would fulfill these goals. Rather, the software would need to be implemented based on two existing foundations:

- In terms of the necessary thermochemical calculations at its core, it would be based on ChemApp [2]. This would not only satisfy the expectations in terms of a stable and fast Gibbs energy minimization engine, but also guarantee that all thermochemical data available in the form of databases for FactSage could be utilized.
- As for the software environment, it became clear that due to the requirement that the resulting simulation should provide a maximum amount of flexibility, modularity, and extensibility, the solution would not be to develop a static program with an interface allowing user-defined code to be added, but in fact to implement a different paradigm: The integration of specialized process simulation components into a state-of-the-art RAD software environment.

The choice quickly fell on Borland Delphi®, not only because ChemApp had already been available for Delphi at that time, but in particular because only a few software development environments were available that fulfilled two requirements which are quite opposed to each other: On the one hand it should provide an environment that is not only easy to learn, but also suitable for users for whom programming is more an infrequent task rather than a daily occupation. On the other hand, the development environment would need to be a state-of-the-art system that also satisfies the requirements of professional software engineers by supporting all aspects of modern programming by design, not just by extension.

3.1. Object-oriented design

SimuSage was implemented as a ChemApp-based, object oriented, extensible class library of both visual and non-visual components for Borland Delphi.

Although SimuSage is based on ChemApp, it uses a different programming paradigm. While ChemApp was developed in FORTRAN at a time when object oriented software development was a technology still in its infancy and rarely used outside university institutes, SimuSage is completely object oriented by design. Only this type of implementation allows for the maximum flexibility and extensibility as experienced by the user, beyond what the SimuSage developers can foresee.

The SimuSage component library presents itself visually as a list of items on a separate tab on the tool palette in Delphi (see Fig. 1). While these *visual* components are the immediate and most common view of how SimuSage presents itself as a tool to the developer, it is only the "tip of the iceberg" in terms of the tools it provides. This design concept is fundamental to object oriented programming, it implements the software building blocks as *classes* which consist of *properties* (i. e. something an object *has or is*) and *methods* (i. e. something an object can *do*). Early on in the development it became clear that this concept is eminently compatible with the application area of process engineering and flowsheeting, where the concept of unit operations and the material streams that are used to link them can be thought of easily as a set of components that are classes with a variety of properties and methods.

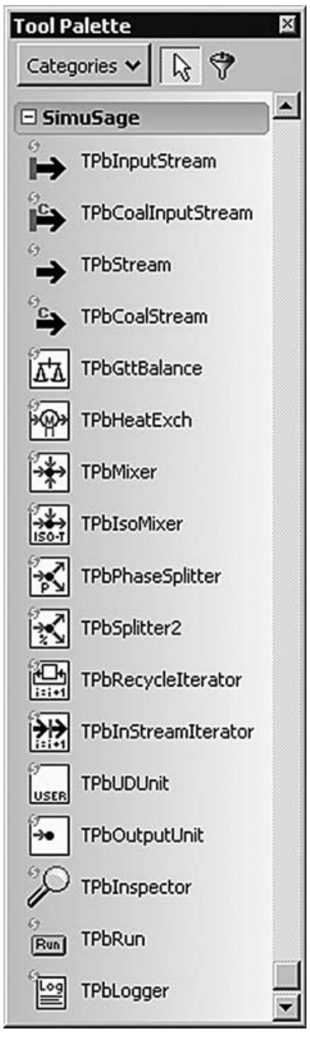


Fig. 1. SimuSage tool bar in Delphi

This concept of *encapsulation* is one of the core concepts of object oriented software design. Other core concepts which are also highly relevant to SimuSage are *inheritance* and *polymorphism*. *Inheritance* allows both the developers of SimuSage as well as its users to add to its library without "reinventing the wheel" for every new class. To take advantage of this, one simply picks an existing, suitable class as a base, automatically inherits all its properties and methods, and only adds new features or changes existing ones. Within the SimuSage component library, the relationship between *streams* and *input streams* is a suitable example. While streams connect unit operations and take care of the transfer of matter between them, a special type of stream is required to model the material input to a flowsheet. Here, a stream needs to be additionally associated with an input material (e. g. air, steel scrap, or lime) in order to model the supply of user-defined material to the process. These input streams are very much like regular streams with a few added properties (e. g. the name of the material with whom they are associated) and methods (e. g. for reading the user-defined composition of this material from a file).

The third important core concept of object oriented software design used in SimuSage, *polymorphism*, is simpler to understand than its name implies. If, for instance, a user develops his or her own unit operation by inheriting from an existing unit operation, polymorphism makes sure that the code in SimuSage that will process the flowsheet when the simulation is executed knows what to do with it, despite the fact that this new custom unit operation is completely unknown to the original developers of SimuSage. While SimuSage was designed and implemented as an object oriented component library, it should be emphasized that this does not mean a new user of SimuSage needs to have previous experience in object oriented programming. On the contrary, for most users of SimuSage this is the first contact with object oriented programming. For simple process simulations, users barely come in contact at all with the underlying object oriented programming principles. On the other hand, demands to increase the complexity of their process simulations will lead them quickly to appreciate how much potential lies even in the modest use of object oriented programming techniques.

3.2. Visual and non-visual components

A typical unit operation in SimuSage, and at the same time the most basic and important one, is the equilibrium reactor. As an object, it has a *visual* representation on the flowsheet, that allows for an interaction both with the programmer during the development stage, but also with the end user at run time, since its visual representation will automatically be a part of the user interface of the resulting program. At the same time, a *visual component* such as an equilibrium reactor consists of *properties* (e. g. its temperature) as well as *methods* (e. g. the capability to calculate its equilibrium). This object oriented concept is not only implemented for all visual components that have a graphical representation on the flowsheet, but also for *non-visual components*. Typical representatives of non-visual components are, for instance, phases. While a phase (e. g. a liquid phase or a stoichiometric compound) usually does not need to have a visual representation on the flowsheet, it still has properties (e. g. an amount, or phase constituents, which in turn are other objects) and methods (e. g. a procedure to copy the contents from another phase).

3.3. The component library

The relationship between a phase and a phase constituent, where both are implemented as classes and the former includes the latter, or the relationship between a stream and an input stream, hints at the fact that SimuSage contains an entire hierarchical tree of classes which form a library of components. Important classes are the unit operations and streams, but much of the versatility of SimuSage comes from the non-visual classes.

3.3.1. Unit operations

For a process simulation tool to be useful in high-temperature inorganic chemistry, for instance in metallurgy or combustion engineering, it does not need to provide an exhaustive list of unit operations such as distillation columns, fabric filters, or pumps. Key is the reliable operation of basic units such as an equilibrium reactor that can handle complex chemical systems with a large number of mixture phases, combined with the ability of the basic unit operations to be extended and customized. Most processes in this application area can be assembled from a limited amount of these basic unit operations. Some of the basic unit operations available in SimuSage include the following:

- **Equilibrium reactors** in SimuSage may be operated at constant temperature, constant volume, or a fixed enthalpy difference. They support the concept of stream states, which allows phases or phase constituents easily to be excluded from the equilibrium calculation on an as-needed basis. After an equilibrium reactor has been calculated in the flowsheet sequence, its equilibrium state is copied to its outgoing stream.
- **Mixers** are available in several modifications to combine two or more streams into one.
- **Splitters** can split a stream according to phases, or according to fractions which can be independently specified down to the level of phase constituents.
- **Heat exchangers** are used to heat or cool a stream by defining either a temperature or an enthalpy difference.
- **Output units** constitute the logical end of a flowsheet. Each flowsheet has one or more output units, which serve as a starting point for the calculation of the flowsheet (see also Section 3.4 below).
- **Iterators** are used to implement user-defined, iterative changes to a flowsheet, or help manage closed loops, which can occur when recycle streams or counter-current processes are modeled.
- **User-defined unit operations** are "empty shells", i. e. the user needs to provide code that processes the ingoing streams. They provide a way to create individual, customized unit operations without the need to use the concept of inheritance.

3.3.2. Streams

Streams represent the flow of material from one unit operation to another, and consist of one or more phases. They are very closely related to the concept of streams in ChemApp, but in SimuSage their object oriented implementation provides added possibilities. The same is valid for the implementation of phases, phase constituents, and system components. Each phase consists of one or more phase constituents. The composition and stoichiometry of phase constituents is expressed using system components.

In addition to the regular stream types, which all consist of phases and their constituents that are physically present in the associated thermochemical data-file, SimuSage supports a separate stream type for condensed fuel streams. The exact compounds of which a specific fuel consists, i. e. the fuel analysis in terms of phases and constituents, are in reality often not known. In contrast, a fuel stream is defined in terms of its chemical composition for instance by specifying its moisture content, the short analysis (ash, volatiles, fixed carbon), the elementary analysis (C, H, N, O – ash free, S), the ash analysis, and its calorific value. Such fuel streams cannot be directly mixed with regular streams, but can be fed together with regular streams into a reactor.

3.3.3. Non-visual classes

Other important, but non-visual classes include for instance phases and phase constituents, system components, stream types and stream states. These non-visual classes, just like the visual classes too, can be accessed as objects in the user's code. They can be declared, created, copied, manipulated, and destroyed much like any other dynamic data type in Delphi.

3.4. Calculational model

Every unit operation has a method named **process** which performs the calculation of the unit operation on the basis of the ingoing streams' values (composition, temperature, pressure, enthalpy), and determines these values for the outgoing stream(s).

Every stream carries a property which is assigned either the status *calculated* or *not calculated*. If, for instance, the supplied amount of a material associated with an input stream is changed by the user, this information of the change is propagated "downstream" through the flowsheet to all the streams and units connected directly and indirectly to the input stream. Thus the status of all streams affected by this change will be set to *not calculated*.

A flowsheet is calculated by calling the method **process** of the flowsheet's output unit. The method **process** is called recursively for all preceding unit operations located upstream whose outgoing streams do not yet have the status *calculated*. The method **process** first updates all of the ingoing streams of a unit, then calls its own method **calculate** to perform the main calculational part of a unit operation, and finally changes the status of its outgoing streams to *calculated*.

Calling the **process** method of a particular unit operation enables the selective recalculation of only parts of the flowsheet.

A simple flowsheet that only consists of unit operations and streams must not contain any cycles and needs to be fully defined. It is considered to be fully defined if all unit operations are joined suitably with streams, i. e. if every stream has a source and a destination unit, every input stream a destination unit, and all units a path to an output unit. If a more complex flowsheet contains closed cycles, as is for instance the case when processes with recycle streams or counter-current multi-reactor processes are modeled, iterators are used. Several types of iterators are available to break closed loops and allow for a custom control of input streams and flow conditions.

3.5. User interface elements

The SimuSage system is completed by a variety of components and visual interface elements that handle the communication with the user, both at design time and at run time. This includes report editors to display contents and properties of unit operations and streams, material editors to define the composition of materials associated with input streams, stream state editors to change the status of phases and phase constituents in reactors, input fields to associate values such as temperature, pressure, or amounts with input streams or unit operations, etc.

A central user interface element is the Inspector, a visual component that supports the user in checking, both at design time and run time, whether a flowsheet contains a number of typical errors such as unconnected streams or undefined essential properties.

3.6. Advanced techniques

Simple flowsheets (i. e. those which do not depend on user-defined unit operations, internal closed loops, or custom output) can be set up very quickly even by users new to SimuSage and without the need to write any code by dragging-and-dropping the necessary unit operations and streams onto the flowsheet and setting their properties interactively through Delphi's dialogs. On the other hand, utilizing advanced techniques provided by SimuSage as well as Delphi allows for the creation of highly sophisticated and customized simulations.

Even without developing user-defined unit operations, SimuSage allows for the customization of the behavior of standard unit operations by using events. In event-based programming the user can typically influence the flow of a program not only by user interaction (mouse clicks or keystrokes), but the program can provide custom events for which the user can write code (event handlers) that is triggered whenever the event occurs. As an example, every unit operation provides a number of events, one of those is named **OnBeforeCalculation** and occurs immediately before the method **calculate** of a unit is executed, another called **OnCalculated** occurs immediately afterwards. By writing an event handler that reacts to these events, the user can for instance customize the behavior of an existing unit operation, or update a display of results once it has been calculated. Other than deriving new unit operations from existing ones by inheritance, which has already been mentioned above, SimuSage also allows for the grouping of components or parts of a network as a "macro". This macro can be reused in other places of the flowsheet, thus reducing the development time necessary to set up complex networks that consist of multiple occurrences of similar parts.

Once a user becomes familiar with programming in Delphi, SimuSage components and flowsheets can be extended, modified, and augmented in almost limitless variety. This also relates to the way the finished simulation displays results, or how a flowsheet presents itself in terms of its user interface to the end user.

One of the features of SimuSage which cannot be matched by any conventional process simulation program is the ability to handle every object it provides in a completely dynamic fashion. All SimuSage components, including visual components such as unit operations and streams, can be created, destroyed, and manipulated through the user code. This enables the developer to add code which not only changes the properties of components, but the assembly of the entire flowsheet at run time. For instance, unit operations can be dynamically created and removed on demand, and streams can be redirected to model a different material flow, irrespective of the initial layout at design time.

4. A simple example

A simple example is meant to illustrate the use of SimuSage. Figure 2 shows a process sketch of a thermal waste treatment process. Sewage sludge is incinerated in a fluidized-bed furnace, for which additional fuel plus the supply of air is necessary. Most of the ash fraction leaves the furnace through the grate, while some of it is carried over as fly ash with the exhaust gas, where it is separated in a hot gas cyclone. The flue gas leaving the cyclone still contains a small fraction of dust filter ash. The process sketch shows the temperatures of the furnace and the cyclone, plus the amounts of input materials.

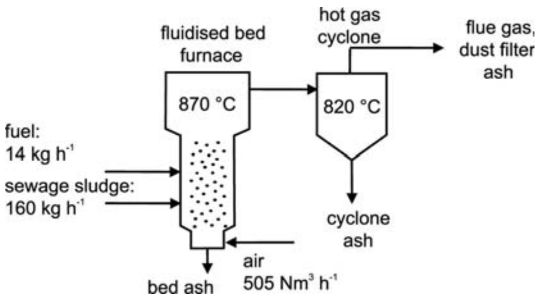


Fig. 2. Process sketch of a simple thermal waste treatment process (Nm^3 indicates normal cubic meters)

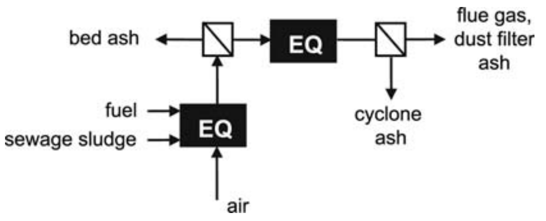
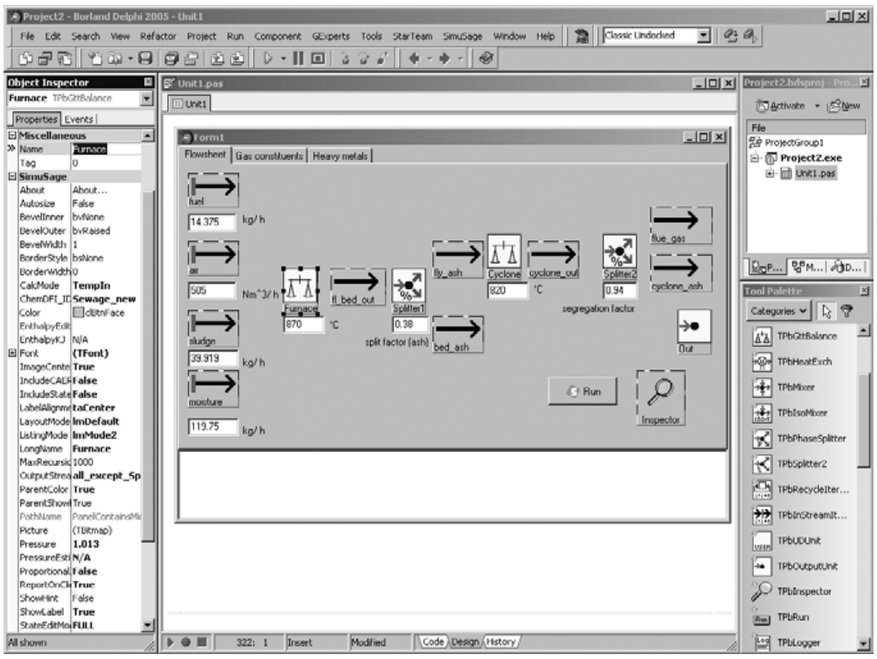


Fig. 3. Process model based on SimuSage components



⊟ Miscellaneous	
Name	Furnace
Tag	0
⊟ SimuSage	
About	About...
Autosize	False
BevelInner	bvNone
BevelOuter	bvRaised
BevelWidth	1
BorderStyle	bsNone
BorderWidth	0
CalcMode	TempIn
ChemDFI_ID	Sewage_new
Color	clBtnFace
EnthalpyEdit	
EnthalpyKJ	N/A
⊞ Font	(TFont)
ImageCente	True
IncludeCALF	False
IncludeState	False
LabelAlignme	taCenter
LayoutMode	lmDefault
ListingMode	lmMode2
LongName	Furnace
MaxRecursio	1000
OutputStrea	all_except_Sp
ParentColor	True
ParentShow	True
PathName	PanelContainsMo
Picture	(TBitmap)
Pressure	1.013
PressureEsti	N/A
Proportional	False
ReportOnCli	True
ShowHint	False
ShowLabel	True
StateEditMo	FULL

All shown

Fig. 4. The SimuSage model during the design stage

For the model, a thermochemical data-file for the 18-component system Pb–Hg–Cd–As–Zn–Cu–Ni–Fe–Cr–Ca–Cl–S–Si–Na–O–N–C–H is extracted from FactSage. The chemical system contains a gas phase with almost 200 constituents, 210 pure compounds, and 26 non-ideal condensed solution phases such as slag, molten salts, and spinel phases.

Based on this data-file, input materials are set up with a dedicated materials editor tool. For both the sewage sludge as well as the fuel an elemental analysis is available that permits these input materials to be defined based on the system components. Air, on the other hand, is defined as a mixture of the gas phase constituents O_2 and N_2.

The next step is the conceptual "translation" of the process sketch into a model using SimuSage components, Fig. 3 shows the initial approach. Two equilibrium reactors are used for the furnace and the cyclone, plus two splitters separating the gas from the condensed phases.

Based on this sketch a SimuSage model can set up. Figure 4 shows a screenshot of Delphi's integrated development environment taken during the design stage of the model.

An example of how user-defined Delphi code is used to modify the standard behavior of SimuSage components can be seen when looking at the source code of the model. Through an input field on the flowsheet the end user can specify global split factors for each of the two splitters. This split factor is normally applied to all phases in the stream entering the splitter. In this particular model, the splitters should apply this factor only to the condensed phases, not to the gas phase. The gas phase of each of the streams entering the splitter is supposed to be directed in its entirety to the streams labeled `fly_ash` and `flue_gas`, respectively. As mentioned above, the concept of events provides the mechanism to easily modify the standard behavior of components. Taking the first splitter (`Splitter1`) as an example, an event handler is defined for its `OnBeforeCalculation` event, which first reads the current value of the split factor from the user input field (`UIF_Split1`) and assigns it to the global split factor (property `SplitFactor`). It then uses the splitter's `SplitFactorP` property to set a phase-specific split factor for the `gas_ideal` phase:

```
procedure   TForm1.Splitter1BeforeCalculation
(sender: TPbUnit);
begin
  Splitter1.SplitFactor := 1.0 – UIF_Split1.Flt;
  Splitter1.SplitFactorP['gas_ideal'] := 1.0;
end;
```

Despite the simplicity of the model, several interesting results can already be gained. As an example, Fig. 5 shows the calculated concentrations of several heavy metals in comparison with measured values.

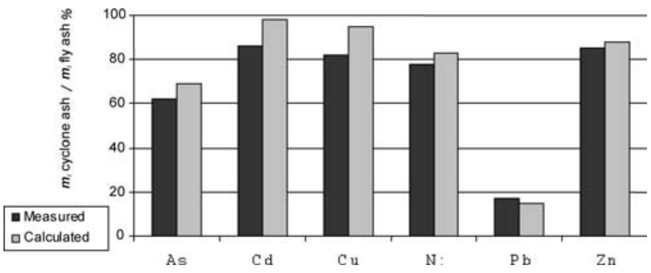


Fig. 5. Sample results of the model – mass ratios of selected heavy metals

5. Applications of SimuSage

The first major application of SimuSage, at the time when it was still under initial development as ProMoSys at SMS Demag and used primarily at IVT, was the simulation of an LD converter process [3–6].

In this steelmaking process, pure oxygen is blown into a molten iron bath for refining purposes. Elements dissolved in the molten iron, most notably C, but also Si, Mn, P and part of the iron are oxidized. This results in a slag phase being formed which covers the hot metal. In the case of carbon, gas bubbles containing CO and CO_2 are formed. In this well documented process, several reaction zones can be identified, which for modeling purposes leads to a division of the entire converter into four basic sections. Each of these sections was modeled based on an equilibrium reactor, and all sections are connected by mass and energy streams according to process conditions. The model was extended by taking into account such effects as stepwise supply of fluxes, energy losses through the converter mouth due to radiation, radiation and convection losses through the converter walls, etc. The results, such as the decarburization rate and the time-dependent amount of elements in the melt were in good agreement with experimental results. The overall excellent results and experiences in applying this new process modeling tool to a non-equilibrium process encouraged the initial developers to continue their work and make it available to other users.

At the same institute, SimuSage was used together with CFD (computational fluid dynamics) models to simulate a biomass fired power plant [7, 8] and a cement plant [9, 10]. In high temperature processes such as cement making and the combustion of biomass fuels, practical problems arise from small amounts of minor components being introduced into the plant via the raw materials. These components tend to form volatile compounds that are transported with the gas phase and form liquid melts when precipitating in cooler regions of the plant. In these regions, dust or fly ash particles, which are present in the gas phase as well, will accumulate on the plant walls and form deposits. In the case of cement making, these deposits will impede material transport in the plant and cause holdups. In the case of biomass combustion, deposits will reduce the heat transfer at the heat exchanger pipes. As accounting for minor component behavior within CFD calculations results in very high computational costs, a combination of CFD and thermochemical simulations was applied to model a cement rotary kiln and a biomass fired grate furnace. In both cases, a CFD calculation is carried out beforehand in order to compute mass and energy transport taking only major chemical reactions into account. From the results of these calculations, SimuSage-based process models were derived which enable the thermochemical conditions to be simulated in greater detail, including the behavior of minor components.

At the Chair of Energy Process Engineering and Thermal Waste Treatment, Technische Universität Bergakademie Freiberg, Germany, SimuSage has been used for a number of projects from the area of power generation involving combustion and gasification processes. Coal-based combined gas and steam cycles allow for the use of coal in power generation at high efficiencies, in particular when pressurized pulverized coal combustion (PPCC) is employed. Hot pressurized flue gas is directly expanded in a gas turbine without prior cooling to exploit the full potential of the high turbine inlet temperatures of modern gas turbines. This necessitates the effective removal of ash particles and corrosive alkali and heavy metals which might otherwise damage the turbine blades. Since the retention of alkali metals in the coal ash is not sufficient, alkali removal using getter materials (e. g. alumosilicates) can be employed to achieve the required limits. To investigate the PPCC process, and especially the potential of an alkali and heavy metal removal using getter materials designed to reduce the risk of sulfate-induced high temperature corrosion, a complex SimuSage-based process model was developed [11, 12].

The condensation of trace elements such as sodium and potassium is also an important concern in the BGL (British Gas – Lurgi) gasification process. The SimuSage model developed consists of several equilibrium stages describing the gasifier zones as well as the slag bath, and permits the simulation of the capture of the volatile ash components by calculating their capture rates and concentrations in the raw gas [13].

Furthermore, at the same institute, SimuSage was used within the framework of the SIMEX project, whose goal was the development of models in order to improve the understanding of ash formation and deposition mechanisms in coal fired boilers. A model for the combustion process was developed which permits the calculation of ash formation in different combustion atmospheres (reducing and oxidizing) depending on the temperature profile in the steam plant [14]. In order to analyze the slagging behavior of different types of Rheinland lignite coal, a SimuSage model was developed in which the steam plant is treated as a network of interlinked equilibrium stages [15]. Through linking the various isothermal equilibrium reactors and through the systematic use of bypass streams, non-equilibrium phenomena, which are for instance caused by incomplete mixing or kinetic inhibitions, could be modeled successfully. The process model allows for the estimation of the slagging and deposition behavior for the range of coal types considered.

At Sasol Technology, South Africa, and the University of Leoben, Austria, SimuSage is currently being used to model two industrial processes [16]. Indirect liquefaction of coal to produce low emission synthetic fuel has gained a lot of prominence globally, with rising oil prices and uncertainties regarding crude oil supply. One of the critical technologies that are employed in coal-to-liquids processes is coal gasification. In the gasification process used by Sasol, coal is converted to synthesis gas using steam and oxygen under pressure. SimuSage is used to model the complex chemistry of this process, looking at issues such as gasification, pyrolysis, combustion, and slagging. In order to model the cement clinker production process, the University of Leoben is using SimuSage as a tool to identify conditions of chemical wear of the refractory lining especially in the rotary kiln. Preliminary results with simple models show the suitability of SimuSage for such complex process layouts. Currently a total process model with the ability of considering reaction kinetics is being developed.

The reduction of carbon dioxide emissions is a major challenge in the energy sector. The requirement to meet the European targets for CO_2 reduction from power plants fired by fossil fuels has led to the consideration of CO_2 sequestration, because the increase of efficiency of power plants has a limited impact on the total CO_2 production. Among others, oxyfuel processes are promising power plant concepts which would allow easy CO_2 removal from the exhaust products. In the OXYCOAL-AC process, developed by RWTH Aachen University, the flue gas is filtered, recirculated, and enriched in oxygen by passing through a ceramic high temperature oxygen membrane. At the Institute of Energy Research, Research Centre Jülich, Germany, SimuSage is being used to develop a thermochemical model to investigate various aspects of this process [17], such as the chemical composition of flue gases, ashes, and condensates in different parts of a future OXYCOAL power plant in order to estimate the risks of high temperature corrosion, problems in ash filtration, and fouling of the membrane.

At the same institute, SimuSage is also used in the development of a model aimed at gaining a better understanding of the transport phenomena occurring in metal halide lamps [18]. Advanced metal halide lamps use discharge vessels made of translucent polycrystalline alumina (PCA). The arc tube contains a molten salt mixture which partially vaporizes under operating conditions. During long-term operation, corrosion occurs due to the interaction between the wall material and the molten salt mixture. A SimuSage-based model has been developed which is used to model thermochemistry and transport issues between the source side, where alumina is dissolved, and the sink side, where it precipitates.

6. Conclusion

Although SimuSage has not yet been available as a commercial product for long, it has already proven itself as a tool that offers significant advantages over other process simulation programs. The ability to combine rigorous, multi-component, multi-phase equilibrium thermochemistry with a state-of-the-art integrated development environment supporting rapid application development (RAD) and maximum extensibility due to its nature as a component library, permits the development of process simulation code which would otherwise be impossible, or at least unfeasible. The concept of dividing a process into a finite number of localized equilibrium reactors, linked by streams into a flowsheeting network, which is already considered the primary modeling paradigm of applying ChemApp, is extended into a highly versatile process simulation tool by the design of a library of reusable and expandable components for use with Borland Delphi. Due to the availability of high quality thermochemical data from programs such as FactSage, SimuSage permits the simulation of complex, industrial processes that exhibit strong non-equilibrium and time-dependent characteristics.

References

[1] C.W. Bale, P. Chartrand, S.A. Degterov, G. Eriksson, K. Hack, R.B. Mahfoud, J. Melançon, A.D. Pelton, S. Petersen: CALPHAD 26 (2002) 189.
[2] S. Petersen, K. Hack: Int. J. Mater. Res. (this issue) 2007.
[3] A. Traebert, M. Modigell, P. Monheim, K. Hack: Scand. J. Metall. 28 (1999) 285.
[4] M. Modigell, A. Traebert, P. Monheim, S. Petersen, U. Pickartz: Comput. Chem. Eng. 25 (2001) 723.
[5] M. Modigell, A. Traebert, P. Monheim, S. Petersen, U. Pickartz, in: S. Pierucci (Ed.), European Symposium on Computer Aided Process Engineering 10, Elsevier (2000) 571.
[6] A. Traebert: Methodik zur Modellierung von Hochtemperaturprozessen. Dissertation, RWTH Aachen, Fakultät für Maschinenwesen, 2001.
[7] M. Modigell, D. Liebig, M. Weng, B. Sunderman, in: 21. Deutscher Flammentag, Cottbus, volume 1750 of VDI-Berichte, VDI-Verlag (2003) 609.
[8] D. Liebig: Gekoppelte Prozess- und Strömungssimulation und ihre Anwendung auf Hochtemperaturprozesse. Dissertation, RWTH Aachen, Fakultät für Maschinenwesen, 2005.
[9] T. Ginsberg, D. Liebig, M. Modigell, K. Hack, S. Yousif, in: L. Puigjaner, A. Espuña (Eds.), European Symposium on Computer Aided Process Engineering 15, Elsevier (2005) 361.
[10] T. Ginsberg, D. Liebig, M. Modigell, K. Hack, S. Yousif: Chemie Ingenieur Technik 78 (2006) 689.
[11] B. Meyer, T. Bause: VGB Power Tech. 85 (2005) 42.
[12] T. Bause: Thermodynamik der Alkalimetall- und Schwermetallabscheidung für die Bedingungen der Druckkohlenstaubfeuerung. Dissertation, Technische Universität Bergakademie Freiberg, Fakultät für Maschinenbau, Verfahrens- und Energietechnik, 2005. Freiberger Forschungshefte A 883.
[13] S. Guhl, P. Brüggemann, A. Jochmann, B. Meyer, in: Proceedings of the 23rd International Pittsburgh Coal Conference, September 25–28, 2006, Pittsburgh, P.A. USA, 2006.
[14] M. Muhammadieh: Thermochemical modeling of the combustion of Rheinland lignite coal. 8th Annual GTT-Technologies Workshop, GTT-Technologies, Herzogenrath, Germany, http://www.gtt-technologies.de/, 2006.
[15] M. Muhammadieh: Beitrag zur Ermittlung des Ansatzpotentials von Braunkohlenaschen in Dampferzeugern. Dissertation, Technische Universität Bergakademie Freiberg, Fakultät für Maschinenbau, Verfahrens- und Energietechnik, 2007.
[16] S. Petersen, J. van Dyk, R. Emler: Process Simulation Applications with SimuSage, CALPHAD XXXVI, May 2007, State College, PA, USA.
[17] C. Weber, E. Yazhenskikh, M. Müller: Thermodynamic Modelling of Ash Related Problem, in the OXYCOAL-AC-Process. Submitted to the 24th International Pittsburgh Coal Conference, Johannesburg, South Africa, 2007.
[18] S. Fischer, B. Huang, U. Niemann, T. Markus: Transport phenomena of aluminium oxide in metal halide lamps. Submitted to the 11th International Symposium on the Science and Technology of Light Sources, Shanghai, China, 2007.

(Received March 30, 2007; accepted August 1, 2007)

Bibliography

DOI 10.3139/146.101552
Int. J. Mat. Res. (formerly Z. Metallkd.)
98 (2007) 10; page 946–953
© Carl Hanser Verlag GmbH & Co. KG
ISSN 1862-5282

Correspondence address

Dr.-Ing. Stephan Petersen
GTT-Technologies
Kaiserstraße 100, D-52134 Herzogenrath, Germany
Tel.: +49 2407 59 533
Fax: +49 2407 59 661
E-mail: sp@gtt-technologies.de

You will find the article and additional material by entering the document number MK101552 on our website at www.ijmr.de