Final Report

Award Number: DE-FC07-00CH11021

Project Title: **Computational Chemistry and Reaction Engineering Workbench**

Principle Investigator: J. Thomas McKinnon
(303) 273-3098 (voice)
(303) 273-3730 (email)
jmckinno@mines.edu

Date: 19 December 2003

Project Period: March 01, 2000 to September 30, 2003

Project Amount: \$1,391,503

Recipient: Colorado School of Mines
Department of Chemical Engineering
Golden, CO 80401

Subcontractors: National Renewable Energy Laboratory, Golden, CO
University of Delaware, Newark, DE
Wesleyan University, Middletown, CT
Reaction Design, Inc., San Diego, CA

Other Partners: Exxon Research and Engineering Company
RES Group, Inc.
Gaussian, Inc.
National Institute of Standards and Technology
Adapco LTD
Dupont
Blue Star Renewable Technologies

II. Executive Summary

The chemical process industries are among the most energy intensive in the US. While much of the energy use cannot be avoided (e.g., feedstock energy or separations costs), there are tremendous gains to be had through simulation-based optimizations. The heart of any chemical operation is the chemical reactor and it is here that gains can be made either directly (e.g., reduced heat flux) or indirectly (e.g., reduced downstream separations cost due to more efficient reactor design). The objective of this project was to develop an integrated suite of software to facilitate the simulation and optimization of chemical reactors. In the phase of the project supported by this grant, we focused specifically on gas-phase chemical processes such as combustion and pyrolysis.

Modeling of gas-phase chemical reactors requires a description of both the chemical processes and the fluid mechanics involved. Current limitations in computer hardware and software preclude handling both these aspects at high fidelity. In our software, called the OpenChem Workbench, we chose to focus our efforts on a highly detailed description of the chemistry while treating the fluid mechanics of the reactors at an idealized level (e.g., plug flow reactors). The chemistry is represented by a large collection of elementary chemical reactions called the mechanism. Elementary reactions represent the actual chemical events that occur at the molecular level.

Before we initiated this project, a large body of simulation software already existed to aid the chemical reaction engineer. However, what was lacking was the *integration* of the existing software. Thus, while in principle a reaction engineer could undertake complex simulations, in practice it rarely happened due to massive inhibiting factors such as incompatible data formats and complex user interfaces.

The implementation of the goals of the project is carried out in three components. First, we collected the necessary software components and wrote our own where necessary. Second, we implemented a graphical user interface to reduce the learning curve for new users and speed the modeling process for experienced users. Finally, we implemented a relational database which shuttled data between the various components, eliminating data format problems.

The final result of this project is the OpenChem Workbench software and its commercial cousin OpenChem Pro. This is a highly sophisticated, but ever evolving, software platform for modeling industrial chemical reactors. Although a new commercial product, it is already making headway in the marketplace.

III. Introduction

In order to meet the challenges of the 21st century, U.S. chemical process industries will increasingly rely on computer modeling to improve process efficiencies, enhance product quality, lower energy consumption, and reduce unwanted emissions into the environment. The “2020 Technology Vision for the Chemical Industry” recognizes computational molecular science and computational fluid dynamics as critical enabling technologies. Reactor modeling in the chemical process industry (CPI) is an important piece of this trend. An essential component of a reactor modeling strategy involves having reliable thermodynamic and kinetic parameters for the compounds involved and robust codes for process simulation. Fortunately, extremely rapid and far-reaching advances have been made in the field of reactor modeling that can assist in the implementation of broad-based modeling strategies to enhance system performance. Currently, a broad array of simulation tools exists to help in the modeling. However, the learning curve required to implement these tools is so steep that their use in industrial modeling is limited to a few specialists or in-house consultants. For U.S. industry to truly take advantage of the rapid technical gains in chemical reactor modeling that have been made in the last two decades, these tools must be made accessible outside the realm of the specialist. That is, empower the process engineer with the tools of modern chemical reactor modeling. The objective of the OpenChem Workbench project was to develop such an integrated chemical reaction engineering modeling program that will be useful to specialists in the field and maintain an ease of use that will benefit process engineers. The program integrates a suite of reaction engineering modules into a unified package. The components of the OpenChem Workbench are:

- (1) thermodynamic property estimation,
- (2) rate constant estimation,
- (3) an equilibrium solver,
- (4) chemical kinetic integrators,
- (5) sensitivity and uncertainty analysis,
- (6) parameter estimation and optimization,
- (7) mechanism reduction, and
- (8) computational fluid dynamics.

The OpenChem Workbench has a modern graphical user interface (GUI) on the front end that dramatically reduces the learning curve as compared to text-based command line programs. A relational database is included on the back end as a searchable repository for data. An expert system provides help and guidance to the user in the selection of the proper computational chemistry method. The three-tier architecture allows for the integration of distributed computing technologies for remote database access and for off-loading challenging problems to a workstation or supercomputer.

The key innovation presented by the OpenChem Workbench is the seamless integration of a large suite of chemical reaction engineering modeling tools. As important as the integration of the tools presented in this document, the OpenChem Workbench also provides an open interface/open source framework for the integration of any reaction engineering modeling tool by the chemical industry.

The secondary innovation afforded by the OpenChem Workbench is integration of distributed computing through a suite of technologies known as web services. Web services hold the promise of truly simplified network computing. That is, running a simulation over the network (on a high-powered server) is as simple as running on a local processor – and totally transparent to the user. The interface technologies behind web services are based on standard internet protocols (e.g., HTTP) and thus become platform and language independent. Web services have been identified as a technology potentially as revolutionary as the Internet itself. We feel it is important that the CPI take advantage of the efficiencies and productivity benefits of web services as the technology develops.

Figure 1 shows a schematic of the components of the OpenChem Workbench. The diagram is the same as that presented in the original proposal. The boxes shown in gray were not implemented due to minor changes in focus over the course of the project.

Figure 1. OpenChem Workbench components. Comparison of originally proposed components (all) to those implemented (white).

The OpenChem Workbench software was developed collaboratively by a large team of reaction engineering experts. The table and text below show the team members and their field of expertise.

Team Member	Company or Organization
Prof. J. Thomas McKinnon	Colorado School of Mines
Dr. Mark R. Nimlos	National Renewable Energy Laboratory
Dr. David Klipstein	Reaction Design, Inc.
Dr. Anthony M. Dean	Exxon Research and Engineering/CSM
Dr. Glen H. Ko	Mitsubishi Chemical/RES Group, Inc.
Dr. Michael Frisch	Gaussian, Inc./Lorentzian, Inc.
Prof. Hai Wang	University of Delaware
Prof. George Petersson	Wesleyan University
Dr. Gary Mallard	NIST

Roles of the team members

- Prof. McKinnon is the Project Manager/Principle Investigator for this project. His primary role is, then, to coordinate and integrate the various tasks. Other roles include oversight of the Workbench Operation Components, Development of the Graphical User Interface, Local Database Development, Report Generation, and Development of Educational Modules.
- Dr. Nimlos brings expertise in quantum mechanical modeling and combustion modeling to the project. He is responsible for oversight of the Workbench Chemistry Components, development of the thermodynamic property estimation methods, distributed computing, and the expert system.
- Dr. Klipstein is the president of Reaction Design, Inc., a company committed to the commercialization of chemical reactor modeling programs. Dr. Klipstein will be in charge of integration of chemical kinetic integrators into the Workbench, implementing sensitivity and uncertainty analysis, and parameter estimation and optimization.
- Dr. Dean has been using the individual tools that will be integrated into the Workbench in the CPI for two decades and, thus, brings a valuable industry perspective to the team. He will be responsible for implementation of the rate constant estimation methods, along with Prof. Wang.
- Dr. Ko brings his experience in process modeling of chemical plants to this project. His primary role will be in problem definition, implementation of the user interface, and local database issues.
- Dr. Frisch is the president of Gaussian, Inc., a company that commercializes state-of-the-art quantum molecular orbital programs. For this project, he will help develop the interface between the Workbench and the molecular orbital programs.
- Prof. Wang brings his talents in quantum chemical modeling and evaluation of rate constants from these calculations to this project. He will be in charge, along with Dr. Dean, of the task

on rate constant estimation methods, in addition he will handle the equilibrium solver and the mechanism reduction methods.

- Prof. Petersson has been developing new molecular orbital quantum chemical modeling methods for over two decades. His role in this project is to implement new methods which can deliver chemically accurate thermodynamic properties for reasonable computer times.
- Dr. Mallard has developed the highly successful WebBook and other databases for the National Institute of Standards and Technology. For this project, he will develop interfaces to allow the Workbench to tap into the databases of the WebBook.

Because the Computational Chemistry and Reaction Engineering Workbench was developed by such a large and geographically distributed team, we felt it was important to develop a **mission statement**:

“The mission of the Workbench Project is to develop a software product in under three years that is useable by individuals with a normal degree of technical training in chemical engineering, chemistry, and related fields but no particular expertise in computational chemistry. The function of the product will be to develop accurate reaction mechanisms and to incorporate them into the simulation of idealized chemical reactors. All required data management tools and utility programs will be available to the user of this program under a single, integrated user interface.”

IV. Project Milestones

In this section we track the 14 tasks from the original proposal. A different perspective of the same information is given in Section IV.6.

Task # and Title	Status	% completed
Task 1. Design of the local database	The local database architecture is entirely completed. The database has been populated with a large amount of molecular and kinetic data.	100%
Task 2. User input → queries for local database.	The user can query the database for molecule and reaction information. The search results can be saved for subsequent uses.	100%
Task 3. User input → queries for remote database.	The user can query the NIST Webbook database and download molecules along with their thermodynamic and structural properties.	100%
Task 4. Integrate the results of common queries.	Query results are saved to the TreeView for subsequent uses (e.g., creating a Set or a	100%

	Stream from saved molecules).	
Task 5. Molecule builder.	The user can build molecules using an intuitive graphical builder tool. The 3-D structure of the created molecules can be optimized at the level of molecular mechanics.	100%
Task 6. Equilibrium interface with the GUI and databases.	The equilibrium module is fully completed with an XML interface.	100%
Task 7. Group additivity	The user can obtain rapid estimates of the thermodynamic properties of molecules using group additivity. A molecule parser automatically determines the groups and looks up the group tables in the database.	100%
Task 9. Graphical user interface.	Done. See screen shots in Section VI.B.	100%
Task 10. Throwaway GUI prototype.	The “throwaway” graphical user interface was a valuable tool in developing the production GUI. This task was completed in Year 1.	100%
Task 11. Gaussian 98 interface.	We have developed a robust G98 post-processor. The input files to Gaussian relatively simple (easy to build by hand), so we have not completed a full G98 integration.	100%
Task 12. Help system	Ongoing improvements to support new features.	100%
Task 13. Uncertainty analysis	No progress. This feature has been removed from the Workbench.	0%
Task 14. Extensible markup language (XML) interface	We now have XML specifications for molecules and reactor models. XML tools (e.g., XSLT) facilitate model upgrades and improvements.	100%

V. Budget

The table below shows the breakdown of the budget year-by-year along with the total for CSM, the subcontractors, and the industrial matching-fund providers. There were no significant variances across the budget and the project was completed within budget.

Final Report
DE-FC07-00CH11021

	Proj. Yr1 - 3/01/00 to 2/28/01			Proj. Yr2 3/01/01 to 2/28/02		
	Projected	Actual	Delta	Projected	Actual	Delta
CSM total	\$ 102.5	\$ 84.0	\$ 18.5	\$ 95.5	\$ 99.8	\$ (4.3)
U Delaware	\$ 72.6	\$ 43.4	\$ 29.2	\$ 77.2	\$ 63.2	\$ 14.0
Wesleyan U	\$ 72.4	\$ 27.5	\$ 44.9	\$ 72.4	\$ 114.2	\$ (41.8)
Reaction Design	\$ 60.0	\$ 57.2	\$ 2.8	\$ 60.0	\$ 40.0	\$ 20.0
NREL	\$ 70.0	\$ 70.0	\$ -	\$ 70.0	\$ 70.0	\$ -
Subcontractor Total	\$ 275.0	\$ 198.1	\$ 76.9	\$ 279.6	\$ 287.4	\$ (7.8)
Overhead	\$ 62.1	\$ 29.6	\$ 32.5	\$ 62.1	\$ 50.2	\$ 11.9
DOE Total	\$ 439.6	\$ 311.7	\$ 127.9	\$ 437.2	\$ 437.4	\$ (0.2)
In-kind Matching Funds						
Reaction Design	\$ 200.0	\$ 173.0	\$ (27.0)	\$ 200.0	\$ 255.0	\$ 55.0
MC-RIC	\$ 160.5	\$ 186.0	\$ 25.5	\$ 160.5	\$ 246.0	\$ 85.5
ExxonMobil	\$ 45.0	\$ 45.0	\$ -	\$ 45.0	\$ 63.0	\$ 18.0
Gaussian	\$ 49.5	\$ 7.0	\$ (42.5)	\$ 18.0	\$ 37.0	\$ 19.0
CSM	\$ 8.3	\$ 11.0	\$ 2.7	\$ 8.3	\$ 6.0	\$ (2.3)
Blue Star	\$ -	\$ 9.0	\$ 9.0	\$ -	\$ 11.0	\$ 11.0
Adapco	\$ -	\$ -	\$ -	\$ -	\$ 7.0	\$ 7.0
DuPont	\$ -	\$ 53.0	\$ 53.0	\$ -	\$ 44.0	\$ 44.0
Match Total	\$ 463.3	\$ 484.0	\$ 20.7	\$ 431.8	\$ 669.0	\$ 237.2
Project Total	\$ 902.9	\$ 795.7	\$ 148.6	\$ 869.0	\$ 913.4	\$ 44.0

	Proj. Yr3 3/01/02 to 9/30/03			Total		
	Projected	Actual	Delta	Projected	Actual	Delta
CSM total	\$ 234.2	\$ 237.9	\$ (3.7)	\$ 432.2	\$ 421.7	\$ 10.5
U Delaware	\$ 54.9	\$ 98.1	\$ (43.2)	\$ 204.7	\$ 204.7	\$ -
Wesleyan U	\$ 51.3	\$ 54.4	\$ (3.1)	\$ 196.1	\$ 196.1	\$ -
Reaction Design	\$ 42.2	\$ 65.0	\$ (22.8)	\$ 162.2	\$ 162.2	\$ -
NREL	\$ 70.0	\$ 70.0	\$ -	\$ 210.0	\$ 210.0	\$ -
Subcontractor Total	\$ 279.6	\$ 287.5	\$ (7.9)	\$ 773.0	\$ 773.0	\$ -
Overhead	\$ 62.1	\$ 117.0	\$ (54.9)	\$ 186.3	\$ 196.8	\$ (10.5)
DOE Total	\$ 514.7	\$ 642.4	\$ (66.5)	\$ 1,391.5	\$ 1,391.5	\$ -
In-kind Matching Funds						
Reaction Design	\$ 200.0	\$ 172.0	\$ (28.0)	\$ 600.0	\$ 600.0	\$ -
MC-RIC	\$ 160.5	\$ 48.0	\$ (112.5)	\$ 481.5	\$ 480.0	\$ (1.5)
ExxonMobil	\$ 45.0	\$ 27.0	\$ (18.0)	\$ 135.0	\$ 135.0	\$ -
Gaussian	\$ 18.0	\$ 41.0	\$ 23.0	\$ 85.5	\$ 85.0	\$ (0.5)
CSM	\$ 8.3	\$ -	\$ (8.3)	\$ 24.9	\$ 17.0	\$ (7.9)
Blue Star	\$ -	\$ 5.0	\$ 5.0	\$ -	\$ 25.0	\$ 25.0
Adapco	\$ -	\$ 43.0	\$ 43.0	\$ -	\$ 50.0	\$ 50.0
DuPont	\$ -	\$ 3.0	\$ 3.0	\$ -	\$ 100.0	\$ 100.0
Match Total	\$ -	\$ 3.0	\$ 3.0	\$ -	\$ 100.0	\$ 100.0
Project Total	\$ 431.8	\$ 339.0	\$ (92.8)	\$ 1,326.9	\$ 1,492.0	\$ 165.1
	\$ 946.5	\$ 981.4	\$ (159.3)	\$ 2,718.4	\$ 2,690.5	\$ 33.3

VI. Technical Report Section

A. Project Description

The OpenChem Workbench software is a large suite which integrates a large number of modules useful to building and testing large reaction mechanisms. The program follows the **object oriented** software architecture throughout. That is, there are software entities (objects) that have a one-to-one correlation to real objects (e.g., molecules) or abstract objects (e.g., reaction mechanisms). In this section we will describe the object architecture of the OpenChem Workbench and the operation of a representative number of the modules.

1. Object Architecture

In this section we will describe some of the objects that make up OCWB.

a. Molecule

We will start with the most basic unit in the OCWB – the molecule. While it may seem quite simple to represent a molecule to a computer, the representation is actually quite complex. First, we need to determine what properties uniquely represent a molecule and, equally important, how can they be represented in a fashion that is in the common language of reaction engineers. The OCWB defines a set of *fundamental properties* for a molecule. The obvious most important fundamental property is *stoichiometry* (or formula) which represents no difficulty in a common language or database representation. The next fundamental property is *connectivity*. Here there is no problem when dealing with human users who can mentally parse a 2-D image of two molecules and decide whether they are the same or different. However, this becomes a somewhat difficult problem for a computer because there is no *canonical numbering* of atoms in a molecule. That is, there is no way to state definitively and uniquely that a given atom is Number 1, another is Number 2, and so on. Fortunately the theoretical mathematicians have found this *graph isomorphism* problem to be intellectually stimulating and we have leveraged their work to our advantage.

The rest of the fundamental properties are charge (integer), cis/trans (controlled list – cis, trans, NA, mixed), spin multiplicity (integer), and natural abundance (Boolean). Some compromises are obvious from this list of selected fundamental properties. For example, it is arguably better to use electronic state rather than spin multiplicity. However, employing this method would violate the rule that the definition and understanding of the fundamental properties must be common among reaction engineers. Electronic states are defined in a complex spectroscopic notation which is not familiar to a wide range of the targeted users of OCWB. For the same reason, we do not use stereo isomers as a fundamental property. This is unfortunate, but there are three separate (and non-complimentary) methods used to define stereo isomers. Furthermore,

these are even less familiar to our intended audience than electronic states. Despite these minor limitations, we have found the definition of molecule fundamental properties to be quite robust for the reaction engineering purposes intended for OCWB.

Once we have a molecule uniquely defined by the computer, along with computational algorithms for storing, searching, and comparing, we need to address the problem that molecules can have multiple properties. Consider the simple example of CH₃OH. One user may refer to this by the molecular formula “CH₃OH”. However, other users may use “CH₄O”, “methanol”, “methyl alcohol”, “wood alcohol” or others. (The NIST Webbook lists 25 names for this molecule!). In OCWB, we declare one name to be a *specifier* for the molecule. This is the main, or default, name for the molecule and must be unique across the database. If the user attempts to save a second molecule to an existing specifier name, OCWB will generate a warning. The additional names are *alternate names*. An unlimited number of these may be used and they need not be unique (we shall see later that they must be unique in a given *thermset*).

The next tier of molecule specifiers represents the thermodynamic properties. Here we have both raw data (enthalpy – H, entropy – S, and heat capacity – C_p) at specified temperatures and what we call the *thermfit*. A thermfit represents a collection of parameters from which the thermodynamic properties can be computed over a specified temperature range. In OCWB we use the 14-parameter set of polynomial coefficients developed by NASA and made popular by Sandia National Laboratories in the Chemkin software. Within OCWB, a thermfit has a thermfit name (which is also saved as an alternate name for the molecule) and a label (a short descriptor that typically defines where the data came from). For example, my thermfit name could be “methanol” and the label could be “88Burcat”. A given molecule can have an unlimited number of thermfits with the only restriction that the concatenation of the “thermfit name” + “label” be unique within the database.

Other properties that can be attached to a molecule are transport data (Lennard-Jones parameters), atomic XYZ coordinates, and atomic-level chirality. Finally, every object within OCWB can have a comment field. The objective is to be able to fully trace any calculation back to the original source of the data through the comments. That is, a reactor simulation will contain the source of the kinetic data for the mechanism. Within the mechanism, the user can then find the source of the thermfit. And from the thermfit the user can find the source of the original data used to generate the polynomial coefficients. Users who have struggled with maintaining this chain of accountability of data manually will recognize the OCWB approach a major step forward.

Figure 2 shows a graphical representation of a simplified molecule object diagram.

Figure 2. The molecule object sub-model (simplified)

b. Reactions, Mechanisms, Projects, and Runs

The brief description of the molecule object model gives a flavor of the issues underlying the development of software objects. Some of the other high-level objects developed for OCWB were: Reactions, Mechanisms, Projects, and Runs. In the interest of brevity the detailed descriptions of these objects will not be given here. Figure 3 shows a higher-level simplified object model.

Figure 3. A high-level simplified object model.

2. Module Implementation

The object architecture, described partly in the text above, is critical to the robust operation of OCWB. However, it is the graphical user interface (GUI) and the modules with which the user interacts. In this section, we describe a representative selection of the OCWB modules.

a. Tree View and High-Level GUI

The OCWB development team made a determined effort to employ the user interface metaphors that are in common use in widely used operating systems (MS Windows, Apple, etc.). We have found a dramatic reduction in the learning curve if we borrow user interface concepts from the set of those with which our users are already familiar. The *Tree View* is a good example. Users are already very familiar with objects (typically files) being represented in a hierarchical structure in tree views. Thus, in OCWB we use the familiar tree view to display and organize our objects – molecules, thermfits/thermsets, reactions, mechanisms, runs, etc. Figure 4 shows a screen shot of the user interface when the user first opens OCWB. While the users may or may not be familiar with reaction engineering, when they first view this GUI they will immediately become comfortable with it. That is, they already know how to navigate the tree, select one or more objects (click, shift-click, or control-click), drag-and-drop, etc. without needing to read a detailed manual to learn these operations. Furthermore, they are presented a familiar-format

Final Report
DE-FC07-00CH11021

pull-down menu along the top and a series of buttons with descriptive icons (and pop-up tool-tips if they are not familiar with the icon).

One of the four Tree Views is always shown in the OCWB GUI. The work panel on the right-hand side of the screen is where the various modules appear as needed. An unlimited number of work panels can be opened (Current Tasks), but only a single one displays. We use the familiar forward/backward arrow metaphor (along with a pull-down menu) to navigate between current tasks.

Figure 4. Tree View screen shot.

b. Set/Stream Editor

Many operations in OCWB require the creation of a collection of molecules and many of these require a collection with a specified concentration and possibly flow rate, temperature and pressure. We define the former as *Sets* and the latter as *Streams*. An example of this operation would be an equilibrium calculation. The inputs to the equilibrium calculation are molecule collections with a specified concentration, or a stream. The user then specifies what molecules are allowed to be in the output but the concentrations are not needed – so a set is specified here. When the equilibrium operation is completed, the output molecules now have mole fractions computed so the set becomes a stream (which can be saved in the database for subsequent operations).

The OCWB provides a Set/Stream editor to create and edit these objects, as shown in Figure 5. From the figure we can see that the Set is specified on the left of the panel and the mole fractions (or mass fractions if desired) are displayed on the right side. OCWB tracks the sum of the mole fractions and will normalize to unity if desired. Each panel has a common set of buttons on the bottom: “Save”, “Cancel”, “Clear”, and “Help”. We can see that the Tree View is now displaying the Project object which contains the sets and streams. Double clicking on an object in the tree brings up the appropriate editor or viewer panel. For operations where there is more than one choice of targets for the object, the user can right click and select from a menu.

Figure 5. Set/Stream editor.

c. Reaction Editor

The user can import reactions from external files (described below) or enter them by hand. Figure 6 shows a screen shot of the Reaction Editor. Now the Tree View displays the Reaction Tree and double clicking on the reaction of interest brings up the editor – with the “Rates” tab being shown here. The reaction is displayed using the names from the specified ThermSet. From this set of thermodynamic properties, the enthalpy and entropy of reaction are computed. The plot shows the rates in an Arrhenius plot format for the forward or reverse reaction. Equilibrium constant plots (K_p or K_c) may also be selected. The “Compare” tab displays the graphical rate information for all the rate parameters selected. This is particularly useful in the case where the reaction may be written with data in both the forward and reverse direction. For example, as set of parameters may be entered for the reaction written as “A + B $\rightarrow$ C + D” and another set for the reaction written as “C + D $\rightarrow$ A + B”. OCWB will allow a consistency check among these data using Compare. Finally, the rate parameters may be edited using the “Edit” tab.

Figure 6. Reaction Editor

d. Molecule Builder

OCWB supports a number of molecule import options, but the user can also build a molecule from scratch. As shown in Figure 7, the user selects the atom and the ligand type desired and drops it into the work area. OCWB takes care of bond angles and makes sure that bonding is homogenous (i.e., a single bond on one ligand is prevented from bonding to a double bond on another). The “+H” button fully hydrogenates a molecule to save time. Molecule zooming and rotating features are supported. A “clean-up” feature fixes the bond lengths and angles into an energy minimization at a molecular mechanics level of theory. The “properties” tab allows all the molecule properties (charge, spin, etc.) to be entered for the new molecule. The molecule can then be saved to the database.

Figure 7. 1-chloroacetaldehyde created in the molecule builder.

e. Import/Export Features

The OCWB supports a large number of importing and exporting options. Since most OCWB users will be migrating from a Chemkin background, we both read and write the Chemkin formats for thermodynamic properties (NASA polynomials), reactions with rate parameters, and transport. We can import molecules directly from the NIST Webbook using either the molecular formula or the name. In this mode, we send an HTML request to NIST for the information. At the NIST server end, it simply appears that a web browser is requesting data. We then parse the returning HTML string and extract the required data. We also support MOL file import and our own XML export/import formatting.

The most challenging part of importing molecules is determining whether or not they exist in the OCWB database. That is, we need to work very hard to make sure that a molecule is not repeated in the database. If it were, it would degrade the ability of OCWB to carry out the valuable comparing operations that the relational database allows. Thus, when a molecule is imported, it falls into one of three categories. Category 1 represents molecules which have no stoichiometry matches in the database. In this case, the molecules are always new and can be saved without any difficulty. Category 3 represents molecules in which the imported molecule (candidate) and a molecule in the database (target) have an exact match for the fundamental properties (formula, connectivity, cis/trans, charge, spin multiplicity, natural abundance). In this case, these save to the database occurs without user input. However, there is a third class – Category 2 – which presents more problems. That is when a molecule matches the formula of a molecule in the database but the candidate molecule does not contain any other fundamental properties. Typically connectivity is the most important one that is missing). Furthermore, in many, many cases there are multiple isomers to choose from or the molecule could be a new one. Since there is no way for OCWB to make the decision without any information, the user must manually match the molecule given a number of choices. Since the molecule imports are generally done with large thermodynamic data sets, the operation can be tedious. OCWB makes the operation somewhat simpler by making default matches when the names match. If there is no name match the default choice is the one with the closest enthalpy of formation.

f. Gaussian Postprocessor

The Gaussian program is the most widely used method of computing thermodynamic properties from quantum chemical calculations. Despite its power, manipulation of the output data is quite tedious. That is, Gaussian generates a huge quantity of data in the form of a large ASCII file. Some of the data are needed but most is chaff that is only required for debugging. In fact, most reaction engineers know very little about what can be done with these data (e.g., molecular orbital coefficients).

OCWB takes the tedium out of this data reduction operation by automatically parsing the Gaussian output and placing the data in the correct fields in the database. One quirk of quantum calculations is that thermodynamic properties are not reported directly. That is, the reference state is isolated atoms rather than the elements. OCWB uses a set of statistical mechanics formulas and a method known as atomization energy to compute the thermodynamic properties in the more familiar engineering units that are useful in reactor design.

g. Rate Builder

OCWB supports several different methods for estimating rate constants from quantum chemical calculations. All these methods require that a transition state (TS) be identified using the Gaussian program. From the TS data and the properties of the reactants and products we can use Transition State Theory (TST) to find the rate parameters in the high pressure limit. However, many reactions are explicitly dependent upon pressure. For this class of reactions we can use two complementary methods: QRRK or RRKM. The former is somewhat less accurate but requires less input data. The computed rate constants are then stored in the database.

h. Mechanism Editor

A mechanism is a large collection of elementary reactions. A typical mechanism may have one thousand or more reactions so the most vexing problem is one of data management. The mechanism editor assists in this task by allowing facile groupings of reactions into logical clusters (e.g., hydrogen abstraction reactions collected together).

Arguably the most useful feature in the entire OCWB is our Mechanism Compare feature. To understand this, let us propose a (very common) scenario in which a researcher on the West Coast (R-WC) and a researcher on the East Coast (R-EC) would like to collaborate. Each has produced their own mechanism for iso-octane combustion and they would like to compare their mechanisms. That is, R-WC wants to know what reactions she has in her mechanism that R-EC does not and vice versa. Where there is overlap they want to compare the rate parameters used. At first blush this seems like a simple task. However, consider the complications.

- 1) Typically, the mechanisms are huge. For C8 combustion, 1000 reactions would be small. Making the Venn intersection of data would be a difficult task manually.
- 2) It is even more difficult because there is no standard naming convention. R-WC uses notations like “yc7h14ooh-z” and R-EC uses notation like “C#CC.C*C”.
- 3) And finally, there is no agreement on the canonical direction in which a reaction should be written. Since all reactions are reversible, it makes no difference to the calculations, but it makes collaborating even more difficult.

Two inherent strengths of OCWB make the compare operation relatively simple. First, we have taken great pains to ensure that a molecule is not entered twice under different specifiers. Second, we have the power of relational databases at our disposal – in particular the searching

capabilities. Thus, we can create a list of reactions that are common to two mechanisms as well as lists of reactions that are unique to one or the other.

i. Reactor Modules

OCWB supports four reactor modules: plug flow reactor (PFR), perfectly stirred reactor (PSR), batch reactor (BATCH), and equilibrium reactor (EQUIL). The reactor modules are integrated to the OCWB as external modules and communicate using the flexible XML specification. Thus, new reactor modules could be added with relatively little difficulty. The reactor modules share a great deal of similarity in terms of the user interface and required information. Each requires that the reactor geometry or time be specified along with the initial stream. EQUIL and PSR generate scalar (zero dimensional) outputs whereas PFR and BATCH generate species profiles that are a function of distance or time, respectively.

OCWB offers a wide array of post-processing analysis. The user can plot concentrations versus the independent variable. To assist making the plots we allow molecule sets to be generated for the purpose of selecting the plotted species. The user can also generate rate-of-production (ROP) plots. In these plots, the user selects a species and all the reactions that create or destroy that species are shown, ranked in decreasing order of maximum flux. The user selects the reactions they want and they are plotted. OCWB also supports a conventional sensitivity analysis where the normalized sensitivity coefficients can be plotted. As with ROP, the reactions are reported in decreasing order of sensitivity coefficient. Finally, we have developed what we call the equilibration parameter (EP). This parameter helps the user determine whether a particular reaction runs in the forward direction (as written), in the backward direction, or is close to equilibrium. The parameter is defined as:

$$EP = \frac{(rate_{forward} - rate_{reverse})}{(rate_{forward} + rate_{reverse})}$$

So, it can be seen that $EP = +1$ when the rate is forward, $EP = -1$ when it is reverse, and $EP = 0$ when the reaction is fully equilibrated. The value of EP is two-fold. First, the reaction engineer tends to think of reactions as always going in the direction as written. If $EP < 0$, then the engineer is alerted to the opposite condition. Second, reactions at or near equilibrium do not show up as contributing much to the sensitivity of any species even if they have a large flux passing through them. Without EP, a reaction engineer may be tempted to delete an important, but equilibrated, reaction.

Figure 8 shows a screen shot of the PFR input panel. As can be seen, the user selects the input stream, the mechanism, and supplies the necessary parameters. Numerical convergence parameters can be specified, if desired, but the defaults almost always work quite well.

Final Report DE-FC07-00CH11021

Figure 8. Screen shot of the PFR module input panel.

B. Project Technology

In developing OCWB, we always strived to employ the best software engineering practices that were appropriate for a project of this scope. In this section we will discuss a number of the methods used. Regarding the general scope of the project, it is useful to collect some statistics:

Lines of code: > 120,000 (this does not include the reactor modules)
 Number of source files: 688
 Number of classes: 880

Using the industry-standard SLOC Count method (<http://www.dwheeler.com/sloccount/>) a project of this magnitude would normally take 30 person-years and cost \$4,000,000. By those metrics, we feel we have provided a good value for the Department of Energy.

The entire program, with a few minor exceptions, is written in the Java programming language. Java allows OCWB to be run on a large number of operating systems with minor changes. At present we support MS Windows and Linux.

1. Use cases

Use cases represent one of the principle design philosophies behind the development of the user interface in OCWB. We basis behind use cases is relatively simple, but powerful – determine how the user will use the program, and what the program will do in response to the user input, as opposed to the opposite. An example of a use case developed for the Reaction Editor is shown in Appendix A.

2. UML

We employed the unified modeling language (UML) to assist in the design of the program classes. UML diagrams help standardize module or plug-in construction and in particular reveals dependencies among the classes and plug-ins in OCWB which enables us to provide a truly modular architecture. Figure 9 shows an example of a UML diagram.

Figure 9. Class diagram for the Bug Reporter plug-in module

3. Three-tier Architecture

We employed what is known as a “three-tier architecture” in OCWB. That is, the program is broken into three distinct layers, or tiers, that can be run all on the same machine or on different machines. The top layer is the graphical user interface, or the client layer. The next tier is the application server which does the computing and “directs traffic”. Finally, the bottom layer is the database server. The advantages of a 3-tier system become obvious when we consider running OCWB in an enterprise-wide environment. The client layer would be the desktop computer of the individual engineer. This computer need not be excessively powerful nor need it be slowed down with complex calculations. The application server can then be a powerful central computer which offloads the complex calculations from the engineer’s desktop. Finally, the data can be stored in a central database that can be accessed by an entire reaction engineering group.

4. Source Code Management

Since we had a large number of developers working on OCWB, we needed a source code management tool to keep on top of revisions and avoid conflicts. The OpenChem Workbench project uses the Concurrent Versioning System (CVS) source code management system (<http://www.cvshome.org/>). This tool consists of a CVS server running on an internet-connected computer; the server manages a repository of source code files. Any file or directory substructure may be “checked out” (i.e., downloaded) by any developer for testing and modification. New and modified files may be “checked in” to the repository. CVS keeps track of the version of each individual file and enables the tagging of any current set of files so that the state of the project can be recalled at a later time. CVS enables developers to share their code, work from different locations, and manage releases of the software. Since CVS is the standard source code management tool it is easy to migrate the project documents from one project hosting provider to another, in particular we were able to move the OpenChem Workbench project from local servers to SourceForge (<http://sourceforge.net/>) with minimal difficulty.

5. Issue/Bug Tracking

Any large software tool such as OpenChem Workbench generates many user responses, including bugs, feature requests, usage questions, and comments. These user responses provide critical information for maintaining and improving the software, so it is a high priority to keep track of these data. To do so we use the JitterBug (<http://samba.anu.edu.au/jitterbug/>) web-based bug tracking system; the OpenChem Workbench site is currently located at <http://gems.mines.edu/cgi-bin/ccrewb>. This tool provides a web page to enter bugs and links to other pages to track their progress. Bugs may also be entered via email. Developers can annotate, organize, respond to, and prioritize the messages via customizable pages.

6. Workbench Help System

The OpenChem Workbench contains a help system to guide the user through operation of the available tools. This system contains HTML-based help pages that are integrated into the workbench with the JavaHelp framework. The help tree is accessed through the OCWB menu and a table of contents and an index allow navigation of the help pages. Context sensitive help is also available to the user through a help button located on most pages. The help system is designed to be modular so that with the addition of new components, a new set of help pages can be added. This will allow easier integration of new software tools in the future.

C. Problems Encountered/Resolution Methods

We ran into two recurring technical challenges in carrying out this project: inflexibility of the database and use-case revisions. While neither of these problems is avoidable, we did learn to deal with the issues as they came up.

1. Database Problems

The power of OCWB to take away most of the tedious work involved in reactor simulations is largely due to the power of the relational database. This is carried out using a complex database architecture with a large number of tables and an even larger number of defined relationships. This power comes at a price, however. That cost is that the database required a large amount of resources to maintain and was painful to modify. Furthermore, we had to provide upgrade pathways for the legacy databases (those developed by OCWB users in the field).

Although this problem will never go away, it can be mitigated through careful planning of the database architecture. That is, think through the use cases in great detail and assess how they will impact the database. If this use cases are carefully crafted, then the number of database revisions will be reduced. The other mitigation step is to recognize that the database will need modifications and design the current version to be “mod-friendly”.

2. Use Case Problems

As we mentioned above, use cases were a very powerful tool for developing the user interface for OCWB. The basic problem is that the job that OCWB is quite complex and the use cases are equally complex. We spent many hours in group meetings debating the merits of various points in the use cases (often in heated discussions). And still, we often had to make a second round of revisions after a GUI has been developed.

There is no real resolution to this problem, but there are two lessons to be learned. First, the time used in developing the use cases is well spent. Even when the meetings seem to be dragging on forever, the public debate of ideas tends to let the best ones filter through. Second, no matter how good the use case is, there will always be problems with it. The developers need to budget for “version 2.0” of almost every GUI.

D. Problems Remaining

OCWB is an ever-evolving software product. The modules described in Section 6.B are largely mature with minimal remaining problems. There are, however, a number of new modules that would increase the value of OCWB. Some of the more important new features will be:

- Tighter integration with CFD programs
- Automatic mechanism reduction
- Parameter estimation
- Expanded reactor coverage (e.g., flame simulator)
- Reactor network support

E. Unexpected or Serendipitous Results

None.

VII. Commercialization Report Section

We have partnered with the RES Group, Inc. of Cambridge, MA to commercialize the OpenChem Workbench. The commercial product name is OpenChem Pro (OCP). At present the RES Group has sold two site-wide licenses for OCP and is close to closing the deal on a third. While the current world-wide economic downturn has made sales difficult, we are encouraged at the positive progress made in such a short amount of time.

A. Status of the Follow-On Funding Commitments

Current sales: LG-Chem, Seoul, Korea. March 2003. US\$80,000.
LG-Caltex, Seoul, Korea. October 2003, US\$20,000.

Projected sales: A major Asian auto maker, to be closed January 2004, US\$77,000.

B. Products and Processes in the Marketplace

As stated above, the product derived from OCWB is the OpenChem Pro software. OCP has more powerful numerical capabilities than OCWB.

C. Revenues Received

Sales: \$100,000.

No other forms of revenue were generated by OCP to date.

D. Efforts at Commercialization

We are aggressively marketing the OCP software along with the chemical consulting services of the RES Group. To date, our most successful marketing and sales has occurred in Asia, but we are working the US and Europe as well.

E. Comparison to the Original Marketing Plan

The marketing plan is essentially the same as that that we made in the proposal approximately four years ago. Perhaps the greatest shift is that we have found the best avenue is to pursue a small number of clients, whose needs we can closely serve, as opposed to selling a large number of licenses with minimal support.

Appendix A: Example Use Case

Reaction Editor Panel

\$Revision: 2.0 \$
\$Date: 2003/11/18 \$
Last modified by \$Author: McKinnon \$

Summary

The Reaction Panel(s) provides the user tools for basic input, output, editing, and manipulation of elementary reactions.

Description

Reactions are implemented using a Reaction tree view and a set of tabbed panes.

Use Cases

1. Enter a new reaction

- a. Select Reaction Editor from the Tools menu, which opens to the “Reaction Summary” tab by default. The user types in a reaction string using “+”, “=”, “→” as delimiters. Hits “enter” or the “Parse” button. Molecules can be entered as the Specifier, the Formula, or an Alternate Name.
 - i. OCP parses the string. A dialog box is brought up to resolve conflicts. When the box is closed, the unique identity of each molecule is known. The names displayed are either Specifier, Formula, or ThermFit name, depending on the choice in the radio buttons.
 - ii. OCP will warn the user if the reaction is not balanced. However, unbalanced reactions can be saved with the Balanced flag set to False.
- b. The order of the Molecules in the Reaction string is preserved in the database. This ordering is attached to a RateParameterSet. That is “A + A + B = C”, “2 A + B = C”, and “B + 2A = C” are all the same Reaction. However, the way the Reaction is displayed depends upon the RateParameterSet selected.

2. Change display names.

- a. The user chooses the way the name will be displayed using radio buttons: Specifier, Formula, or ThermFit name. The name display can be changed at any time.

- i. If a ThermSet is not specified and ThermFit name is selected, OCP requests that one be selected.

3. Create a new reaction from an existing one

- a. The user R-clicks (or D-clicks) on a Reaction in the Tree and selects Reaction Editor. The user then types in the changes to the text box. If the changes are merely “+ M” or “(+ M)”, then the actual Reaction is the same, but a new RateParameterSet will be applied. If the editing changes the molecules, then the Reaction is entirely new. In the latter case, all the RateParameterSets attached to the original Reaction are removed from the display.
- b. The user can Save the new Reaction with or without a RateParameterSet. (it isn’t very important to save a Reaction without a RateParameterSet, so if this creates problems then ignore it).

4. Adding new rate parameters

- a. The user types in a ParameterSetName (labeled “Rate Param Set Name” if there is room for such a long label), selects the class of rate parameters (Arrhenius, etc.), and enters the numbers.
- b. The rate parameters can be Saved.

5. Editing rate parameters.

- a. The user selects an existing parameter set and types in new numbers over the old.
- b. The name can be new or remain unchanged.
- c. If the name does not change, OCP will warn if any Runs will be invalidated.

6. Pressure regimes.

- a. OCP will recognize the pressure regime depending upon the way the string is typed in: no “M” = high-pressure limit, “+ M” = low-pressure limit, “(+ M)” = fall-off. The “Third Body Enhancement Factors” table will be disabled if the RateParameterSet is in the high-P limit.
- b. Similarly, if the user enters Troe, Lindemann, or SRI rate parameters, “(+ M)” will automatically be appended to the Reaction string if they are not already there.
- c. If there is a conflict between the string and the parameters (e.g., the user types in “(+ M)” but selects Arrhenius parameters) then OCP should warn the user (if this is too hard, then forget it).

7. Reaction and RateParameterSet descriptors.

- a. OCP will compute and display Forward and Reverse Molecularity.
- b. OCP will compute and display Forward and Reverse Reaction Order, except in the case of Falloff where the definition is ambiguous.
- c. OCP will compute and display Balanced.

- d. The user will select from the following Reaction Types:
 - i. Unspecified (default).
 - ii. Abstraction.
 - iii. Beta-scission.
 - iv. Bond cleavage.
 - v. Recombination.
 - vi. Disproportionation.