SoftLab : A Virtual Laboratory Framework for Computational Science

Ann C. Catlin, Margaret Gaitatzes, Zidu Ma, Elias Houstis, John R. Rice, Nien-Hwa Wang, Shahani Markus, and Sanjiva Weerawarana

Abstract— **The new economic realities require the rapid prototyping of manufactured artifacts and rapid solutions to problems with numerous interrelated elements. In the SoftLab project, we investigate the issues involved in the design and implementation of a virtual laboratory that simulates and controls the functionality of a wet/dry prototyping laboratory. The 21st century prototyping laboratory environments will include: i) integrated view of the design, simulation, manufacturing, and training phases of the underlying prototyping process, ii) virtual environments capable of controlling both the experimental (physical) and computational (simulation) prototypes together with their interactions, iii) knowledge based systems for handling the increasing amount of knowledge and data a desginer must synthesize, compare, trade-off, and optimize, iv) 3-D numerical simulation systems that support the different possible models of the underlying physical process, v) virtual computational environments that exploit high performance computing (HPC) and multimedia technologies towards the goal of eliminating or minimizing the need for physical prototypes and construction aids. Creating a prototyping laboratory with such design goals presents a challenge in itself as well as a number of research opportunities. In this paper we present a software framework for a virtual laboratory and report on our experiences in designing and implementing two software prototyping laboratories, microelectronics material (MicroSoftLab) and bioseparation (BioSoftLab) chemical engineering laboratories.**

Keywords— **virtual laboratory, virtual instruments, problem-solving environment (PSE), hybrid PSE, design, prototyping, computational science.**

I. Introduction to SoftLab

The fields of computational science and computational engineering force us to address the challenge of integrating scientific instrumentation, computational tools, numerical software, and scientific visualization tools, with the goal of creating very high level systems that target a certain class of problems. Such systems require a wide range of expertise plus a flexible and diverse array of equipment. The SoftLab framework[1] will provide the infrastructure and facilities that serve the needs for basic research broadly focused on building problem-solving environments (PSEs) for applied, experimental work in computational science. The facilities include:

- High-performance graphics processors to support scientific visualization, geometric modeling and design, and multimedia graphical user interfaces for parallel programming and programming in the large.
- High-performance computers.
- Software that allows application researchers to make full use of these facilities without becoming deeply involved in implementation details.
- Virtual laboratories that provide scientists with a full range of functionality, both for experimental activities and for the mathematical modeling and simulation of the experimental processes.

The laboratory presents an ideal environment in which to accept and meet the challenges of computational science and engineering. Issues that must be addressed include mathematical software, electronic prototyping, geometric modeling, parallel algorithms, databases, software engineering, and computer systems. This facility is a catalyst for significant research in computational science and engineering, both within a Computer Science department, as well as in other departments. It accelerates the maturation of a number of collaborative, interdisciplinary research projects that are beginning to create a critical mass of applications-oriented research and expertise in computational science.

The SoftLab project addresses the full range of issues in the future paradigm of computational science, and includes collaboration with researchers in chemical engineering, mechanical engineering, biology, and biochemistry. In this paper, we describe the design and implementation of two prototype virtual laboratories, BioSoftLab and MicroSoftLab. Each is a hybrid problem-solving environment which supports the seamless and intelligent interaction of the experimental and computational models in a single, unified environment. We show how these synthesized environments attempt to fulfill the SoftLab research objective.

II. The SoftLab Virtual Laboratories

At the highest level, SoftLab acts as a gateway to the virtual laboratories that have been implemented using the SoftLab design philosophy. A virtual laboratory is a hybrid PSE which encompasses all research activities that occur in the scientific laboratory environment. It is a software layer above the experimental and computational processes which generally exist in the laboratory side-by-side. SoftLab unites the experimental and computational models in a way that allows them to interact with each other, so that feedback from one can enhance and improve the methodologies applied in the other.

The functional objectives of the virtual laboratory created by SoftLab are the following: support for interacting with remote instruments which control the experimental process, support for simulation of the experimental process (including accessing and executing the PDE code which models the process), incorporation of a training simulator

This work was supported in part by NSF award CCR 9202536 and ARPA ARO award DAAH04-94-G-0010.

for the scientific methodology, and visualization of scientific output data resulting from either research model. Since the software layer must handle the experimental and computational models in a unified way, they must first be coupled from the theoretical and operational standpoints. The input and output associated with both models must appear in a uniform way to the scientist using a virtual lab. It is not usually the case, a priori, that the models are coupled in this manner. Generally, the computational code that simulates the experiment is a Fortran program driven by an input file, where the input data bears little resemblance to the experimental input and experimental process that it represents. In addition, when the experimental input or process is varied, it is often unclear how to modify the computational input data to reflect the changes. The first phase in the implementation of a virtual lab is the off-line process of coupling the experimental and computational models. The primary task is to determine how the setup of the phyical experiment corresponds to the input of the simulation code. For example: how is the selection of specific chemical solutions represented as input to the computational code, how are machine lever and switch settings related to the input data, etc. When the setup of the experiment is sufficiently related to the variables which determine how the computational code is initialized, the design of the virtual laboratory can begin.

In order to support the various activities that take place in a physical laboratory, scientists must be able to use the virtual laboratory to

- perform physical experiments by interacting with and controlling remote instruments,
- simulate physical experiments through computer modeling of the experimental process,
- compare physical and simulated data by playing back recorded experiments, and
- train scientists to understand the physical experiment, the computational model of the experiment, and the relationships between them.

A choice of these four scenarios is available to scientists after they have selected their virtual laboratory at the SoftLab gateway. The graphical interface which is presented is a software representation of the physical laboratory. Each important physical device is present, in particular, all instruments and equipment used during the experimental process must have a visual representation in the virtual laboratory. The corresponding software devices are henceforth called virtual instruments.

Each scenario has a specific functionality that relates to a basic activity occurring in the lab. The first scenario is the *physical experimentation scenario.* Scientists can control the laboratory instruments and experimental process from a remote location, and they can monitor the experiment via animation of the virtual instruments. The animation may be driven by state reports from the physical devices, or by calculations based on the input data. They can control data collection and extraction remotely during the experiment, and they can visualize the experimental results at its conclusion.The virtual lab can also act as an "experiment database", where the experiment's input configuration and output results are saved for later analysis. The option of naming and saving experiments is truly a benefit for the scientists, since information defining the experiments is now automatically available on-line to browse through, catalog, and analyze.

The second scenario is the *virtual experimentation scenario.* Scientists set up the virtual instruments and experimental process just as in the physical scenario, and then the physical setup information is transformed to the input required for the computational model. Numerical parameters (e.g., time steps, mesh specification, tolerance) are defined via special interfaces, and an expert system is on hand to query for process characteristics or computational parameters. During the processing of the computational code, the virtual instruments receive intermediate results so that they can be animated to show the progression of the simulation process. Results can be visualized exactly as in the physical case. The simulated experiment may also be saved to the experiment database. Here, too, the input configuration and output results are used to define the experiment in the database. Since the physical and simulated experiments now reside together, computations on their associated data can easily be done for comparison or analysis.

The third scenario is the *experiment playback scenario* where physical and simulated experiments can be retrieved from the experiment database. Scientists can review or compare experiment input configurations and selectively display output results. Physical and simulated experiments can be compared or combined, and multiple data sets can be visualized concurrently. Physical results can be used to improve the computational model via feedback processes, and simulation results can be used to design new physical experiments. Finally, scientists can "playback" an experiment on the virtual instruments. The functionality of the playback scenario links the two research models in new and important ways, and thus fulfills one of the main objectives of the SoftLab project.

The final scenario is the *training scenario* which can be used by both engineering students and research scientists. It can be used by students to learn about the physical layout of the laboratory, the instrument characteristics and operation, and the experimental methodology. Students can learn about the computational model associated with the experimental process, and how the experimental process and its simulation interact. This scenario can also be used by more experienced researchers to learn how SoftLab can be used as a virtual laboratory. The eventual use of SoftLab's virtual laboratories in engineering classrooms is a major objective of the SoftLab project.

III. The Software Infrastructure of SoftLab

An essential kernel function of the SoftLab laboratory is the representation and operation of the laboratory instruments in the virtual environment. Instruments look like and operate just as their physical counterparts. Each scenario presents this virtual laboratory with working vir-

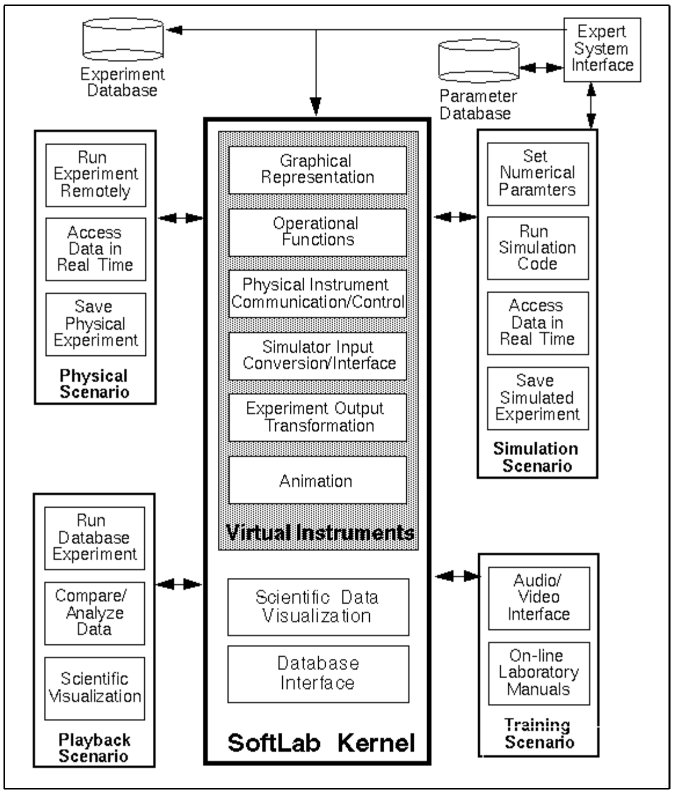

Fig. 1. Functional view of the SoftLab architecture.

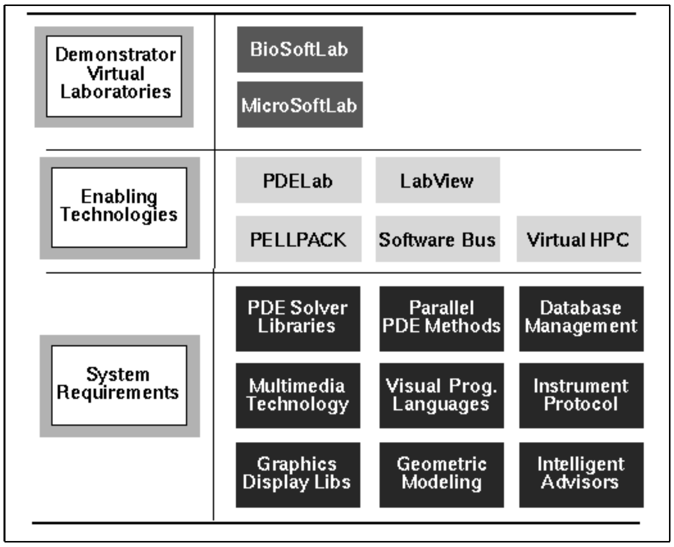

Fig. 2. Layered view of the SoftLab architecture.

tual instruments. These virtual instruments can be programmed for running a physical experiment, simulating the physical process, playing back an experiment, or even for training in the operation of the instrument. For simulations, the virtual laboratory finds and executes the code that represents the appropriate PDE model of the process. During a physical experiment, the virtual laboratory communicates with and controls the remote physcial instruments. In all cases, the instruments are animated to show the progression of the process, physical or simulated.

Access to an experiment database for storage, retrieval, and analysis of experiments is another kernel operation in the virtual lab. Since these functions are required throughout all four scenarios, a common database interface is provided. Some functionality belongs to a restricted subset of the scenarios, such as visualizing results in real time. Other functionality is specific to a single scenario, such as the multi-media annotations used in the training scenario. The specialized components are implemented as building blocks with specific input and output, and the components can be wired together and interact with each other according to the requirements of the scenario.The operation of each scenario is thus determined by the functional components that have been wired together. This also guarantees that the look-and-feel of the environment is uniform across all scenarios. A diagram of the functional components that support the four SoftLab scenarios is shown in Figure 1.

To construct the building blocks for a virtual laboratory with the functionality described above, a kernel of software parts is required. The current realization of this kernel is shown in Figure 2 as a multi-layered software architecture. At the base we have the low-level building blocks which support both the physical laboratory view and the PDE view (computational model) of the experimental process. In order to formulate PDE problems, the necessary tools and facilities include geometric modeling, symbolic processing and and intelligent advisors. Solving PDE problems requires libraries of generalized and domain specific solvers. For the laboratory processes considered for SoftLab implementation, the PDE models cannot usually be solved using general methods. They rely instead on customized code. Since the performace of such code is often measured in days, it is essential to introduce high performance (parallel) computing methods to speed the computations. Graphics display systems are needed for visualization and animation of the solutions and performance data.

To support the remote control of physical instruments, a specialized visual programming language is needed to build the visual representatin of the laboratory instruments and to introduce functionality into the "knobs and levers". In addition, the instrument-specific protocols needed for communication and control must be accessible. Other low-level functionality is required for a complete implementation of the four scenarios: a database management system to save and retrieve experiments, multimedia technology to support audio/visual training and audio/visual communication with the remote laboratory, etc.

The algorithm, language, and system infrastructure demands for building such an environment are great, but the existence of the enabling technologies listed in the next level allows the SoftLab virtual laboratories to be designed and implemented at a meta-level. Most of the low-level functionality is present in one or more of the problem solving environments (PSEs) in the second level.

PDELab[5] and PELLPACK[6] provide the PDE problem solving environements for specifying and solving the PDE process model, and then visualizing the solutions. These PSEs are designed to enable users to solve PDE problems. For example, most domain specific (customized) solvers can be integrated into PELLPACK quite easily. This allows users of the customized code to take advantage of the symbolic processing, geometry builders, mesh generators, and visualizers that are available within PELLPACK. Much effort in the PELLPACK PSE is also devoted to techniques for parallelizing sequential code. These tools and facilities already exist for the solving the PDE models that represent the experimental process. The LabView[2] environment is used to build the visual representation of the virtual instruments and give them "working parts". The LabView visual programming environment is robust and complete; it was used to build all the necessary working instruments for the two demonstration laboratories, BioSoftLab and MicroSoftLab. Communication protocols for some instruments can be obtained through LabView. For those that are not (as was the case for BioSoftLab), the conversions from the virtual instruement user input to the physical instrument input must be done manually. This can only be accomplished when the instrument protocol is released by the manufacturer of the instrument.

When various tools and facilities within the PSEs need to share information, or when the virtual laboratory needs to communicate with the PDEs (or visa versa), the software bus described in [9] is used. The software bus provides the mechanism wherein tools (or facilities) inter-operate together without having explicit knowledge about each other. It also provides the infrastructure necessary for managing a set of distributed tools. The software bus is already a functional component of PDELab, but we needed to apply it to the sharing of information between the environment governing input/output for the virtual instruments and the environment handling the PDE process model associated with that I/O.

No existing tool covered the multimedia requirements of our scenarios, so a new multimedia component, SoftMedia, was built to support SoftLab. This component is now available as an "existing technology" (multimedia PSE) for future virtual laboratories and even more general applications.

The remainder of this paper is as follows: a detailed description of the implementation of the virtual environment for the Bioseparation Laboratory, a brief description of the implementation of the Microelectronics Laboratory, a summary of the functionality of SoftMedia and the conclusion.

IV. BioSoftLab

A. *The physical laboratory and experimental process*

The Bioseparation laboratory is part of Purdue's School of Chemical Engineering. The physical laboratory is shown in Figure 3.

Bioseparation is the process of separating chemical components by passing solution mixtures through an adsorbent column. The column is packed with sorbents selected for their adsorption properties with respect to the incoming solutions. Solutions pass through the column and elude from the bottom. Since each solution component adsorbs to the surface of the sorbents in a unique way, components elude at different times. Reactions within the column may also produce new components. The primary goal of bioseparation is to control what happens in the bioseparation column, so that one can predict when and at what concentration certain components will elude from the column. This process is used for final purification of proteins, chemicals, and biochemicals used in the manufacture of pharmaceutical and food products, for water treatment, and for many other biochemical processes[3].

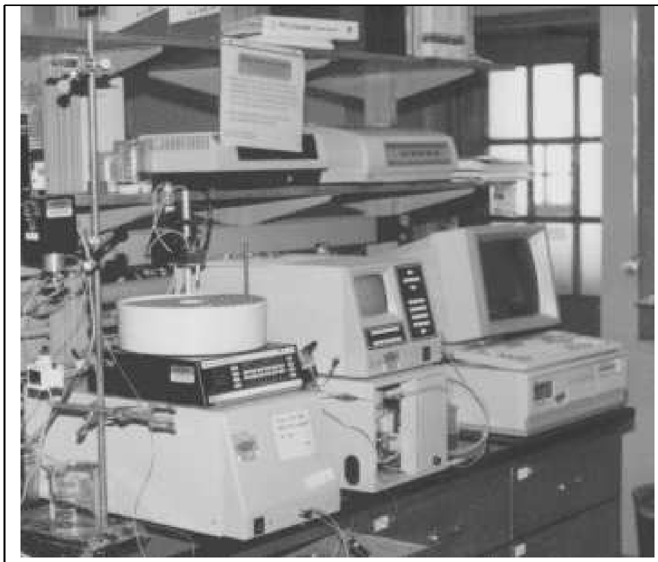

Fig. 3. The bioseparation laboratory.

The physical instruments required to support this research are from the Waters Chromatography Division of Millipore. The bioseparation column is the focal point of the experimental process. Solution mixtures enter the column from two sources. One source is an injection syringe mounted on a tube inserted in the column. The other is a Waters 600E pump controller which regulates flow into the column from a combination of four attached reservoirs. When solutions elude from the column, component concentrations are measured by the Waters photodiode array detector. The measurements are collected by an attached NEC 80286 computer which Waters provides with custom hardware and software for communicating with the detector.

Because bioseparation experiments are both expensive and complex, computer simulation of the process is an attractive alternative. VERSE is a custom implementation of a numerical simulator which models the process, and this code is used by bioseparation scientists to simulate experiments[4].

A.1 A virtual laboratory for research, education and practice

BioSoftLab is designed to merge the two environments, experimental and computational, into one common interface. Before BioSoftLab, the experimental process and the corresponding simulation were separate and independent activities; there was no substantive link between these two

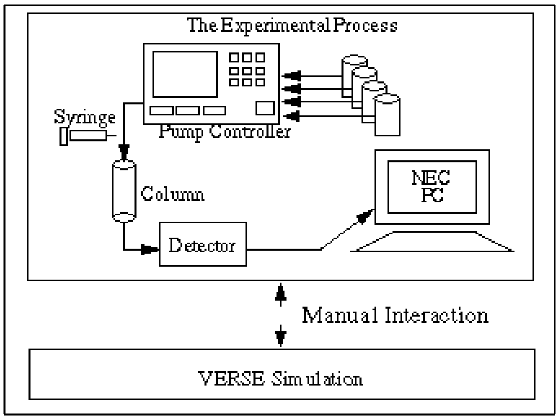

Fig. 4. Diagram of the experimental process and its relationship with the simulation before SoftLab.

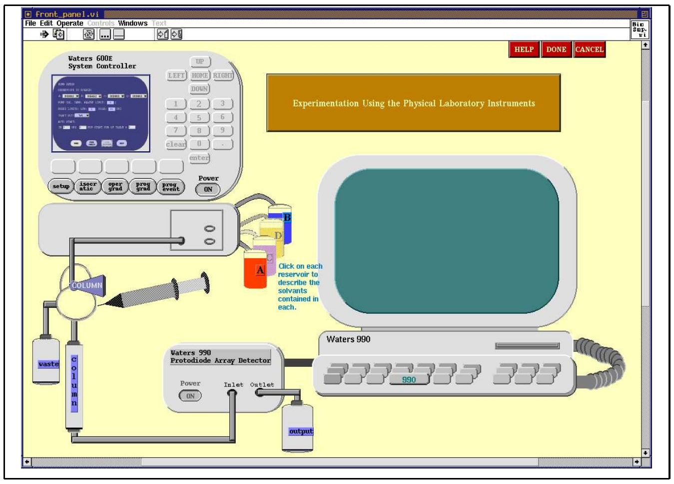

Fig. 5. Entering the physical experimentation scenario.

scenarios. Input to the simulation was completely unrelated to the setup of the experiment which it represented. No automated method existed for comparing experimental output to simulation output, and any procedures for transferring or feeding back information from one scenario to the other were manual (see Figure 4.)

To better understand and implement this common interface, we had to acquire a plethora of information relating the experimental process and the simulation process. By grouping actions together and relating them to the virtual instruments, the bioseparation scientist can now see how different parameters of the simulation input relate to the physical input. This correlation, which becomes obvious in later sections, was not obvious prior to the development of BioSoftLab. During the implementation, it became clear that the new environment could offer more than a common interface to the two environments, it could offer a means of communication between them: data could be passed back and forth and information could be stored for later access and comparison. To support this functionality, the playback and analysis scenario was added.

A.2 Using BioSoftLab to perform experiments

To run a physical experiment using the BioSoftLab interface to the physical lab, one selects the Physical Experiment scenario from the BioSoftLab main menu. BioSoftLab presents the software view of the physical lab and its equipment shown in Figure 5. The functionality of each virtual instrument is identical to that of its physical counterpart.

To set up and run a physical experiment from BioSoftLab, the user must first place the solutions in the reservoirs and syringe in the real laboratory. Afterwards, the user can perform the remainder of the set up remotely using the BioSoftLab interface to the physical laboratory. There are various types of information required by BioSoftLab:

1. *Enabling the equipment.* Enabling the detector and the controller, clearing the loop, etc.
2. *Programming the equipment.* Setting up the (virtual) controller in the same way one would program the physical Waters 600E Controller in the lab, selecting the operating mode of the (virtual) controller, specifying the flow rate of each solvent for each time step, etc.
3. *Setting up the PC for data acquisition.* Running the data acquisition software on the (virtual) PC, specifying how the data should be gathered during the experiment.

This information is identical to the information the user would input to the actual equipment in the physical lab in order to set up a physical experiment. Given this input, BioSoftLab has all the information required to run a physical experiment. After the experiment is started, BioSoftLab displays an animation of what is happening in the physical lab: the reservoirs empty, liquid flows through the column, etc. Currently, this animation information is calculated given the controller flow rate information. In the future, we would like to communicate with the controller to acquire this information and display it in the same way on the BioSoftLab virtual instruments.

When the experiment is complete, BioSoftLab can connect to the Waters PC to access the data collected by the NEC computer. This data is transitted to BioSoftLab and displayed on its virtual PC. After reviewing the data, the user can choose to record the experiment in an experiment database for later retrieval. The information saved in the database includes both the experimental input and output. For the information to be complete (so that we can re-create the experiment or compare it to some other experiment) we ask the user to provide the following information:

1. *Description of the physical makeup of the input solutions.* In order to save the experiment in the database, information like the quantity of each solution used and a description of the chemical composition of each solution is required to fully describe the experiment.
2. *Physical description of the equipment used.* Equipment like the tube connecting the column to the controller and the detector are non-standard, and different tubes could result in different data being generated. We request that the user provide all information relative to the tube, so that re-creating the exact ex-

periment is possible.

The above data is information that an experimental scientist would need to record (in a notebook) to identify or re-create an experiment in the physical lab. Therefore, the information is also required here to fully identify an experiment within the database. In this way, BioSoftLab's database also becomes an "electronic notebook" or experiment depository.

During the implementation of this scenario, we encountered various problems in passing information between the controller and BioSoftLab because the Waters equipment was not designed to allow for remote control. Therefore, control via the virtual instruments was possible only by translating the user's input to the actual controller language. The ease of implementing a physical scenario, therefore, is directly related to the design and functionality of the actual physical equipment. Another problem was more specific to interaction with the user. The virtual environment must be able to request the necessary information in such a way that even novice bioseparation scientists could supply the required input. The interaction between physical equipment is automatic, and we had to simulate the same inter-connection. If the user inputs a certain piece of information then we should be able to derive from it as much as possible, without asking the user to supply more than is absolutely necessary.

A.3 Using BioSoftLab to simulate experiments

In this scenario, the experimental process and the computational model which represents it are linked together and interact with each other in a truly unified manner. To simulate an experiment numerically using the VERSE simulation code, one selects BioSoftLab's virtual mode. BioSoftLab presents you with a software view of the physical laboratory and its equipment (Note that the view presented in Figure 6 is identical to the view of the physical scenario presented in Figure 5). The instruments are set up and programmed exactly as if for a physical experiment; the functionality of each virtual instrument is identical to that of its physical counterpart. However, in the virtual mode, the physical input is then transformed to the input required by the computational model.

Since there are additional numerical parameters and system characteristics associated with simulation, special interfaces built into BioSoftLab's virtual mode allow one to provide the following additional information:

- specification of the physical makeup and characteristics of input solutions and sorbents,
- properties of the input solutions needed by the computational model,
- identification of the reactions (if any) amongst the solutions, that will occur in the column, and
- selection of the process model and associated numerical parameters.

With the physical and numerical input complete, the VERSE simulator is started. Simulation output is collected in real time, and intermediate and final results generated by the simulation can be visually represented on BioSoftLab's virtual instruments. Data can be saved in various formats, and can be loaded into PDELab for three-dimensional scientific visualization and animation. When the simulation has finished, BioSoftLab allows one to "save" the experiment to an experiment database. Saving a virtual experiment means that the input configuration and output results are stored so that they may later be retrieved to playback the experiment or to view intermediate and final results.

In order to implement the above scenario, we encountered some major difficulties. We had already built the software representation of the physical instruments, and these virtual instruments were fully "functional". However, there was no mapping between input/setup for the physical experiment and input to the computational model which was used to simulate it. We had to identify and sep-

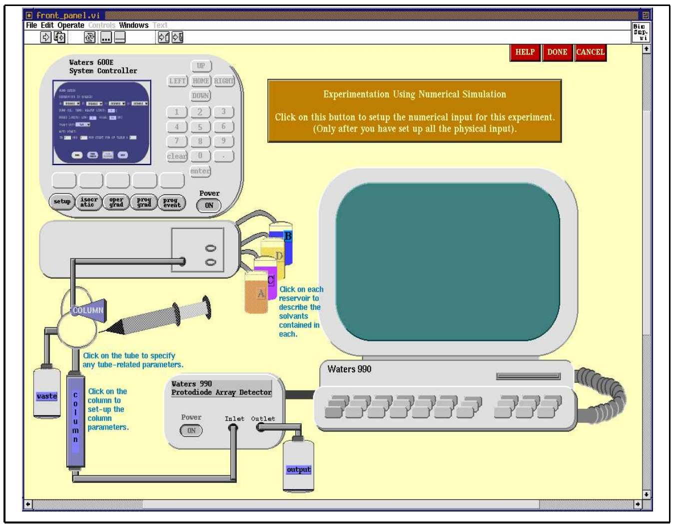

Fig. 6. Entering the virtual experimentation scenario.

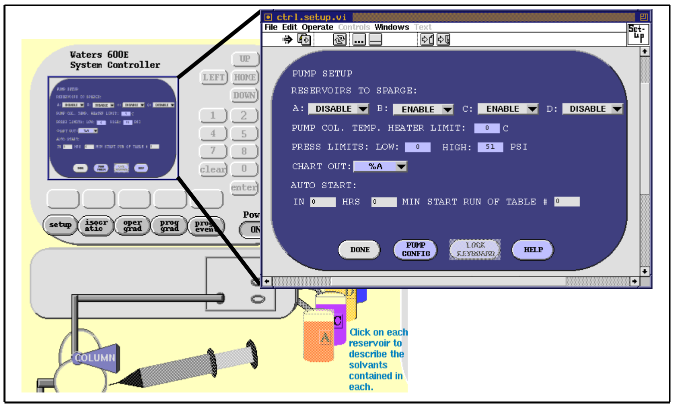

Fig. 7. Instrument setup: programming the virtual pump controller

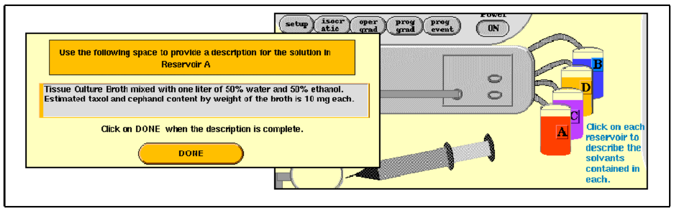

Fig. 8. Solution input: filling reservoir A with tissue culture broth

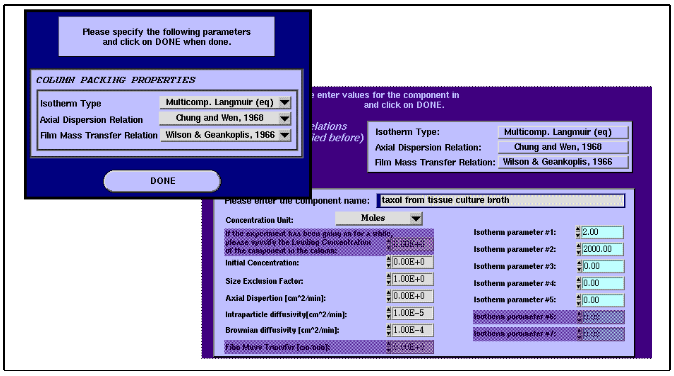

Fig. 9. Component properties: describing taxol numerically.

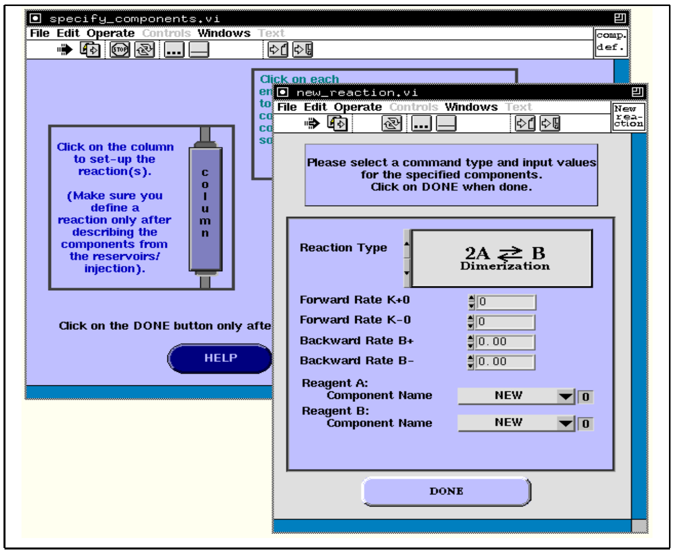

Fig. 10. Reaction specification: identifying dimerization.

arate the kinds of information required by the simulation code. A clear understanding of the differences between the following categories of data enabled us to devise a stable and consistent mapping.

- *Information related to the experimental procedure and instrument setup* .This information should be entered exactly as it is in the physical experimentation scenario (Figure 7.)
- *Information related to the physical makeup and characteristics of the input solutions and sorbents.* Here, the simulation setup must depart from the physical setup. In the physical case, the experiment is driven by the characteristics, properties and makeup of the solutions and sorbents themselves. In the simulated experiment, the physical makeup and properties must be described numerically to the simulator. Therefore, users must specify the contents of the solutions and their physical characteristics. If a reservoir is filled with a "taxol broth" solution, the user must name (or tag) the solution and describe exactly what is contained therein (Figure 8.) Clearly, users should not be required to identify the contents of tagged input more than once.Therefore, this information would take advantage of a database.
- *Information related to the properties of the input solutions which are used to generate numerical parameters.* There are several different examples of this type of data. The most important is the specification of which components from the input solutions are to be simulated. These are special components whose journey through the column is to be monitored by the simulation process (Figure 9.) Special information about these components is needed (for example, isotherm coefficients, diffusivity, etc.). This again requires a database interface for storing and retrieving such data.
- *Reaction information.* Reactions occurring in the column require particular attention. These are considered "components to be monitored", and the component information described above is needed here as well (Figure 10.)
- *Process model specification with associated numerical parameters.* This refers to the system of partial differential equations that will be solved by the simulator, along with the numerical parameters required by the Finite Element Method (FEM) which is used to solve them. There are several "experiments" or "computational models" within VERSE, and the application of a given model is governed by the type of experiment which is being simulated. Ideally, expert system advice for the proper process model should be available. Other FEM parameters, such as the mesh specification, should be tied very closely to the column - since the simulation actually represents what occurs in the column during the experiment. To emphasize the relationship between the physical and simulated experiments, the numerical parameters are tied wherever possible to the virtual bioseparation column .
- *Specification of monitoring and output requirements.* The simulation produces concentration measurements of the specified components over time. This data can be generated for concentrations at the end of the column or at specified points along the length and radius of the column.

Probing the depths of the relationships between the physical process and the computational model required many refinements to our virtual environment. Since the theoretical model allows far greater flexibility in the specification of input than the experiment itself, we had to extend the concept of physical input to allow the experimenter to scale-up, scale-down, optimize, mix or separate the input solutions in ways that were not feasible in the physical laboratory. Since, however, these parameters represented extrapolations of actual experimental input, their values could be specificed on the virtual device used for "actual" input (by extending the range of values, increasing the scale, etc.). In this way, users of the numerical simulation model are able to understand how each parameter of the theoretical input was associated with the actual experimental process.

There are also differences in the way physical and virtual experiments are visualized in BioSoftLab. The simulated experiment is monitored and visualized directly on the Bi-

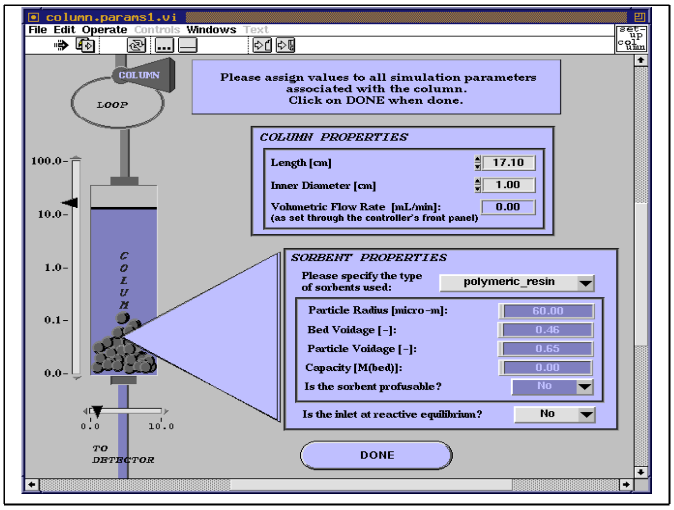

Fig. 11. Packing the column.

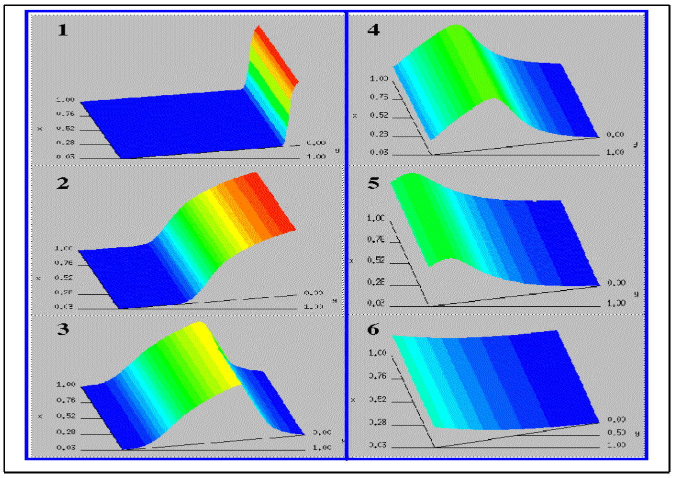

Fig. 12. 3D view of a reactant's concentration as it moves down the column. Here, the concentration is uniform across the column width.

oLabView instruments, just as in the physical experiment. But in the computational case, the data used to "animate" the instruments is generated by the simulation instead of by the actual devices. This again allows the experimenter greater flexibility in visualizing the experiment, since the simulation generates intermediate data which the physical devices cannot produce. Column profiles generated by VERSE produced three-dimensional time-dependent component concentration data along the length of the column, which we visualized on special column-graphs within BioSoftLab. We can also animate this data using scientific visualization tools in PDELab. Graphs such as the ones shown in Figure 12 were not possible before the advent of BioSoftLab.

A.4 Using BioSoftLab to playback experiments

The playback tool is designed to enable scientists to re-run experiments for which the input configuration and the output data already exist. BioSoftLab users could already run both the physical and the virtual scenarios for the same experiment within the same environment. It seemed an obvious extension to provide them with the ability to re-run experiments, either simply to review the input and generated output, or even to compare the input and output of the physical experiment with that of a virtual run. In order to be able to re-run experiments, we constructed a database where bioseparation experiments could be stored and then defined a format for the stored data. As a result, the user can now choose to save an experiment from either the physical or virtual modes. After the user specifies the input to the experiment, BioSoftLab collects all the information necessary to identify the experiment and records it in a file with predefined format. This includes physical characteristics of the solutions, physical dimensions of the column and the tube, etc. This data file is stored in the database where it can be retrieved at a later time. If the user is performing a virtual experiment, the data file generated by BioSoftLab (and used as input to the simulation module, VERSE) is also stored in the database.

Running the physical experiment or the simulated experiment generates various mode-dependent data files. In the case of a physical experiment, if the user opts to retrieve the detector-generated data from the PC, a file showing the concentration at the end of the column is the only file output during the experiment. In the case of a simulated experiment, the user specifies whether a history file (time-dependent end of column concentration) and/or a profile file (time-dependent concentration along the entire length of the column) should to be generated by VERSE, as well as whether PDELab format files should be generated. These output files are all stored into the database together with the input file.

With all the files required to define an experiment now stored within the database, the user is able to re-play an experiment. Selecting the Playback mode from the BioSoftLab menu allows a user to playback a physical experiment, playback a simulation, or compare physical and simulated data.

After the user selects an experiment for playback, BioSoftLab displays the experiment input: physical and/or virtual. The input for the virtual experiment is displayed in a format similar to that of the physical input, except that information about the components and reactions is also included. If the user is simply reviewing the output data corresponding to a virtual scenario, the column profile data is displayed (if it has previously been saved). Next, in both the physical and the virtual scenarios, an animated histogram of the concentration at the end of the column is displayed on the virtual PC monitor. This animation uses the time-stamped output data produced by the experiment. In the hybrid case, when the user is comparing physical data to the simulation results, both the detector-generated histogram and the corresponding VERSE-generated histogram are displayed on the virtual PC so that the results can be compared.

A.5 Using BioSoftLab for bioseparation training

BioSoftLab's training capabilty targets a wide spectrum of users, from novice bioseparation students to biosepara-

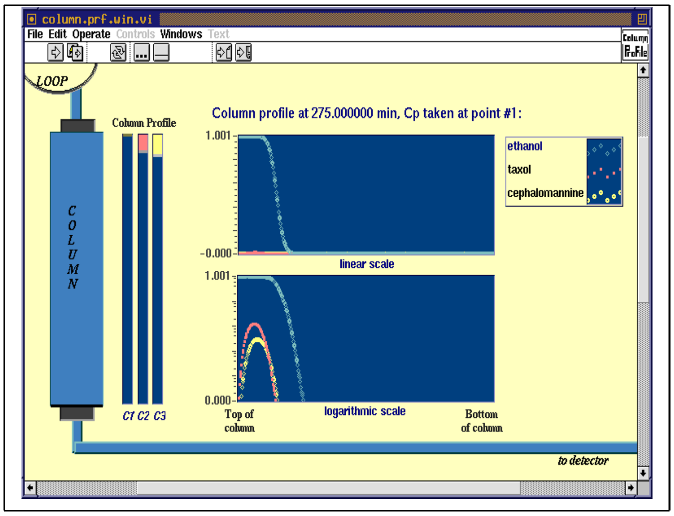

Fig. 13. Monitoring the column during simulation

tion research scientists. It can be used to

- learn about the lab setup and instrument control,
- understand the actual bioseparation process,
- find out how BioSoftLab can be used to setup and "run" a physical bioseparation experiment, and
- learn how to use BioSoftLab to run the computational model.

A bioseparation student can request information about

- The bioseparation process. Various multimedia annotations are employed to give the user an understanding of what a bioseparation experiment involves and what it may be used for.
- The bioseparation laboratory. A description of the instruments used to perform a bioseparation experiment is presented.

Bioseparation scientists can find out how to use BioSoftLab to perform a physical, virtual, or hybrid experiment. When one of these is selected, a guided tour of setting up and running an experiment using BioSoftLab is presented. Using a wide range of multi-media annotations, the user is taught the step-by-step routine required to set up an experiment using BioSoftLab. A user who is already familiar with the actual bioseparation experiment can use BioSoftLab's Teaching Tool to practice setting up a physical or virtual experiment and to better understand the interaction between the physical and computational models.

Clearly BioSoftLab can be used in an engineering classroom as an instructional aid for teaching the physical layout of the laboratory, the equipment characteristics, operation and functionality, safety in the laboratory, experimental methodology, including how to monitor the devices and collect results, use of the computational model to simulate an experiment, and the interaction between the experimental process and its simulation. All of the above can be done from within the classroom, without entering the laboratory where the experiment actually takes place. The advantages of having this capability in the classroom are obvious. It increases expertise in laboratory instrument operation on a large scale and without use of the costly lab. It allows access to laboratories that may not be locally available. It allows beginners to perform "dangerous" or "expensive" experiments. It uses a combination of recorded actual and simulation data to understand how these two models interact. It uses simulation to visualize and explain phenomena not measurable by real-time experimentation.

V. MicoSoftLab

A. The physical laboratory and experimental process

The Microelectronics Laboratory is part of Purdue's School of Chemical Engineering. The experiment involves the chemical vapor deposition (CVD) of silicon from silane on Si(100) substrate surfaces (wafers). In situ infrared spectra emissions are collected in real-time to characterize the silicon surface and study reaction intermediates during the growth of a thin film of Si(100) at 750 degrees C and 10.0 Torr[7].

The microelectronics laboratory consists of a reactor chamber (cell) that is placed inside a Nicolet 800 FTIR spectrometer. The original cell was modified to allow for better temperature control vacuum capabilities. Auxiliary equipment needed to supply the emission cell with N2, SiH4 and cooling water along with all temperature, pressure and mass flow instruments remain on the outside of the spectrometer[8].

B. A virtual laboratory

MicroSoftlab is designed in the same way as its BioSoftlab equivalent. A virtual instrument corresponds to each physical instrument, and the user can operate these instruments to remotely control the experiment. Since the actual experiment requires specifying information for the FTIR, through the Nicolet, every 10 to 15 minutes, MicroSoftlab provides the microelectronics scientist with the ability to preprogram most of this information. MicroSoftlab calculates the required data from the specified input and supplies the Nicolet with the necessary information as needed so the microelectronics scientist need not be present in the lab throughout the entire process.

Only part of the design for MicroSoftLab has been implemented. Remaining tasks include the integration of the computational model and animation of the virtual instruments.

VI. SoftMedia

An essential component of SoftLab is the availability of multimedia functionality. The physical and virtual laboratories are separated from each other (perhaps by many miles), and scientists using the virtual laboratory environment may need to see what is happening in the physical lab. They may also need to communicate with scientists and technicians who are in the lab, and this communication may require video, audio or transmissions of diagrams and computations.

SoftMedia is a general framework for integrating multimedia conferencing tools to any problem solving environment requiring multimedia facilities. An instance of

SoftMedia is a collaborative platform which supports audio conferencing, video conferencing, and whiteboard capabilities. It supports dynamic re-configurability, so that a single interface can be presented to the user across machines with varying low-level audio and video software. These off-the-shelf multi-media tools are used to configure SoftMedia properly, according to the communication requirements of the PSE. SoftMedia is integrated into Softlab, and provides a multi-media platform for collaboration between the physical and virtual laboratories. This allows users of Softlab to look at and monitor the physical laboratory via video conferencing while running a remote experiment using SoftLab's virtual laboratory. Scientists using the virtual environment can also communicate with experimental scientists in the physical lab using audio conferencing and whiteboard facilities.

VII. Conclusions

The traditional prototyping approach involves the definition and construction of a physical system in terms of a descriptive set of specifications and properties followed by the construction of simplified numerical models for simulating the behaviors of that system under some set(s) of physical laws. This process often involves the use of diverse and separate techniques and tools. The user is often forced to manage multiple sets of data and computational environments, making the computational analysis "after-the-fact" rather than an integral part of the prototyping process. This approach is cumbersome, inefficient, restricted in scope, inflexible (since it does not allow the scientist or engineer to easily explore alternative formulations and designs), and impractical to use as training environment. The SoftLab framework allows the seamless integration of physical (experimental) and simulated (computational) prototyping which is controlled by a single virtual environment with a user interface that can be adapted to the targeted physical process. We have demonstrated proof-of-concept in a chemical engineering laboratory associated with the bioseparation process. A SoftLab framework is currently under development for a microelectronics materials chemical engineering laboratory.

References

[1] Christoph M. Hoffman, Elias N. Houstis, John R. Rice, Ann C. Catlin, Margaret Gaitatzes, and Sanjiva Weerawarana, "SoftLab - A virtual laboratory for computational science," *Mathematics and Computers in Simulation 36*, 1994, pp. 479-491.

[2] National Instruments Coporation, LabView User Manual, Autin, Texas, 1992

[3] K. E. Van Cott, R. D. Whitley and N.-H. L. Wang, "Effects of temperature and flow rate on frontal and elution chromatography of aggregating systems", *Separat. Tech nol.1*, 1991, pp. 142-152.

[4] J. Berninger, R. D. Whitley, X. Zhang, N.-H. L. Wang, "A Verstile Model for Simulation of Reaction and Nonequilibrium Dynamics in Multicomponent Fixed-Bed Adsorption Processes", *Computers in Chemical Engineering 15* , 1991, pp. 749-768.

[5] Sanjiva Weerawarana, Elias N. Houstis, John R. Rice, Ann C. Catlin, Cheryl L. Crabill, Chi C. Chui and Shahani Markus, "PDELab: An Object-Oriented Framework for Building Problem Solving Environments for PDE Based Applications", *Proceedings of the 2nd Annual Object-Oriented Numerics Conference*, 1994, pp 79-92.

[6] Elias N. Houstis, John R. Rice, Ann C. Catlin, Sanjiva Weerawarana, and Margaret Gaitatzes, "PELLPACK: A Problem-Solving Environment for PDE-based Applications on Multicomputer Platforms", *ACM TOMS*, to appear.

[7] I-H. Oh and C.G. Takoudis, "Modeling of Epitaxial Silicon Growth from the SiH2Cl2-H2-HCl System in an RF-Heated Pancake Reactor", *J. Appl. Phys. 69*, 1991, pp. 8336-8345.

[8] I-H. Oh and C.G. Takoudis, "Modeling of Chemical Vapor Depostion in Pancake Reactor Systems", *J. Electrochem. Soc., 90-12*, 1990, pp. 75-81.

[9] Sanjiva Weerawarana, Elias N. Houstis, John R. Rice, Ann C. Catlin, Margaret Gaitatzes, Cheryl L. Crabill, Shahani Markus, and Tzvetan Drashanshy "The Purdue PSE Kernel: Towards a Kernel for Building PSEs", *CDS 96-082*, 1996, pp 1-11.