SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (in press)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.817

S^4W: a problem-solving environment for wireless system design

Dhananjay Mishra[1], Clifford A. Shaffer[1,*,†], Naren Ramakrishnan[1], Layne T. Watson[1,2], Kyung K. Bae[3], Jian He[1], Alex A. Verstak[1] and William H. Tranter[3]

[1]*Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.*
[2]*Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.*
[3]*Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.*

SUMMARY

This work describes the Site-Specific System Simulator for Wireless System Design (S^4W), a problem-solving environment (PSE) that integrates visualization and computational tools with a high-level graphical user interface. S^4W improves the ability of wireless system engineers to design an indoor wireless system by encouraging them to think in terms of designing the system for optimal performance. Issues of computation management, data management, and location of resources are hidden from the user. The complex nature of data sets in the domain of wireless simulations calls for a customized set of visualization tools. Therefore, a number of *ad hoc* visualizations were developed for S^4W. A study comparing the integrated system with an earlier, unintegrated version is presented. This helps to demonstrate the productivity gains that a PSE provides. Copyright © 2007 John Wiley & Sons, Ltd.

Received 25 May 2005; Revised 5 January 2007; Accepted 12 January 2007

KEY WORDS: problem-solving environment; wireless system design; visualization; database-centered architecture

*Correspondence to: Clifford A. Shaffer, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.
†E-mail: shaffer@vt.edu

Contract/grant sponsor: National Science Foundation; contract/grant numbers: EIA-9974956, EIA-9984317 and EIA-0103660

1. INTRODUCTION

A problem-solving environment (PSE) is an integrated system that provides a convenient set of high-level tools that allows users to define and modify a problem description, choose solution strategies, manage software and hardware resources, visualize and analyze results, and record and coordinate problem-solving tasks. They are intended to provide all the necessary tools to the user for solving some scientific problem [1]. Gallopoulos *et al.* [2] suggests: 'The time required for many design, analysis, and production tasks will be shortened by one or two orders of magnitude. Further, the results produced will in many cases be better and more reliable. The challenge is to encapsulate our problem-solving knowhow into easily used, flexible systems.' Our own experience comparing an integrated PSE with a functionally equivalent but disparate collection of tools bears this out.

While PSEs in computational science are often complex, have a rich software architecture, and integrate large-scale simulations, the only part of a PSE visible to the user is the user interface. The user interface's goal is to keep the user from worrying about underlying intricacies, while still providing the required functionality.

The Site-Specific System Simulator for Wireless System Design (S^4W) is a PSE developed at Virginia Tech (Blacksburg, VA). The term 'site specific' refers to a communications channel in which the performance characteristics of the channel are strongly dependent on the physical locations and the environment that surrounds the transmitter and receiver. For example, in a ray-tracer-based simulation, an explicit geometric model of the site is part of the problem specification. This is in contrast to, for example, a simple channel model that predicts signal strength based solely on the distance between the transmitter and receiver, which might be appropriate for computing the power received in space.

The key functionality of S^4W is to predict the performance of wireless systems in specific physical environments. S^4W uses both deterministic electromagnetic propagation and stochastic wireless system channel models as underlying simulation components. S^4W supports running simulations and visualizing results, validating simulators by comparison with field measurements, optimizing placements of transmitters and receivers, and managing the results produced by experiments [3].

S^4W has two simulators: a site-specific propagation model based on ray tracing, and a Monte Carlo simulation for wideband code division multiple access (WCDMA) [4]. Ray tracing is a popular high-fidelity deterministic model for wireless signal propagation. The ray tracer predicts the impulse response of a wireless channel. S^4W also includes a global optimizer based on the DIRECT algorithm of Jones *et al.* [5].

A major design goal for S^4W was to integrate visualization tools with the PSE and control the whole system from one unified interface. Visualization is the key to understanding the results from large simulations in computational science and engineering. Sometimes visualizations may be generated by a generic tool, but more often *ad hoc* visualization tools are needed [6].

Feedback during runtime is useful for deciding whether a computation is performing as intended. Based on visual feedback, a user should be able to stop and re-run the computation with different parameters. This feature is incorporated in the design for simulation runs using S^4W.

In addition to its unified interface, the other major architectural design feature for S^4W is the central role played by the database system. All problem descriptions and solution results are stored in the database. Much of what the user does in S^4W directly relates to browsing, managing, and modifying the contents of the database.

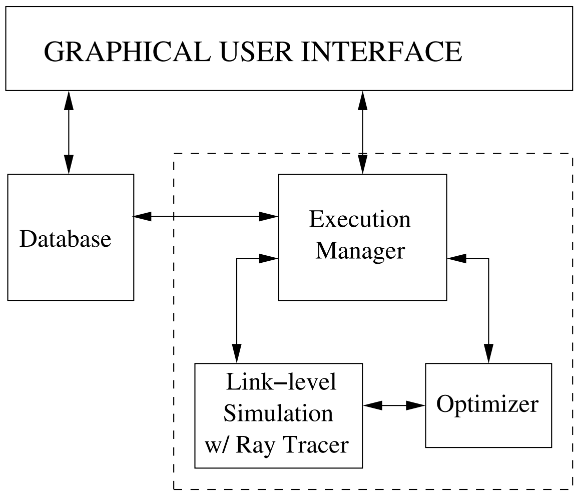


Figure 1. Outline of the S^{4}W design. Components inside the dashed line form the computational unit of the PSE. Computational components communicate with the GUI via a database.

S^{4}W can serve the needs of at least three classes of users. The first consists of researchers designing receivers. Such users must deal with the patterns of delay profiles in the power received. The ray tracer simulation provided by S^{4}W allows wireless system designers to see the power delay profiles under various conditions, so that receivers can be designed based on these power delay profiles. The second class of users is channel model designers who need an environment in which to test their simulators [3]. S^{4}W allows users to conveniently compare actual measurements of an environment with the output from the simulator for their channel model. The third class of potential users comprises field engineers who must site transmitters and receivers in a complex building environment.

It is said 'the whole is greater than sum of the parts', and the synergy between the parts is key to the success of PSEs. Prior to implementing the integrated GUI for S^{4}W, visualizations were available in some form for all of our components. However, the idea of a PSE entails (a) putting them in context, (b) adding features pertinent to the nature of the problem domain, and (c) integrating with the user interface. A sketch of the GUI's interaction with the other components via a central database is shown in Figure 1. A comparison of the integrated system with an earlier prototype that contained the same simulation and optimization capabilities but without any integration is presented in Section 5. More details concerning all aspects of the system can be found in [7].

2. RELATED WORK

Many problem-specific PSEs have been developed for various application domains. Our review is focused on work regarding interfaces and visualizations provided by PSEs.

PSEs have come a long way [2] from ELLPACK [8] for solving two-dimensional (2D) and three-dimensional (3D) elliptic partial differential equations. Geographically distributed and Web-based PSEs, such as WEB-IS, are being used in the geosciences for remote data analysis and visualization. WEB-IS promises to bring the user interface to handheld devices connected to the Internet via a wireless connection [9]. Another example of a Web-based PSE is Web PELLPACK, which allows the user to define a partial differential equation problem, choose and configure solution strategies, and then visualize and analyze the results [10]. Many recent PSEs involve multiple platforms or the use of multiple programming languages and distributed components over the network [11]. Network-based PSEs are envisioned as providing network-based 'software services' [12]. Virtual Cell [13] is another example of a Web-centric PSE. Similar to S^{4}W, Virtual Cell also has a database system playing a central role.

From simpler domains such as PDEs [8,10] and linear algebra [14], PSEs have migrated into diverse applications such as wood-based composite design (WBCSim [15,16]), bioinformatics and computational biology (Expresso [17], H++ [18], JigCell [19], and Virtual Cell [13]), gas turbine dynamics simulation [20], and conceptual aircraft design (VizCraft [21]).

Some of the earliest known PSEs such as ELLPACK [8] did not have a GUI. However, interactive ELLPACK [22] added a GUI to a later version. Later still, the parallel version of ELLPACK (called PELLPACK [10]) had a more sophisticated and portable user interface. These three phases of ELLPACK span a period of 13 years. The creators of PELLPACK define a PSE as having a user interface, libraries, a knowledge base, and a 'software bus' for transferring data between them. This definition also includes software systems such as Matlab, Mathematica, and Maple. PELLPACK defined the user interface as consisting of a high-level language and graphical interface that allows the user to specify the problem and visualize the solution in some 'natural' form.

A number of PSEs integrate analysis codes with optimization methods in a flexible manner, along with a GUI for reviewing the results of an optimization. Framework for Inter-Disciplinary Optimization (FIDO) [23] demonstrates distributed and parallel execution using an integrated user interface. FIDO provides the ability to modify input parameters while the application is executing. iSIGHT [24] provides a generic shell environment for multidisciplinary optimization. LMS Optimus [25] provides a user interface in order to set up a problem, select from a number of predefined methods to be used with the problem, and analyze the results. The DAKOTA toolkit [26] provides a flexible, object-oriented, and extensible PSE with an integrated interface for a variety of optimization methods. VizCraft [21] provides a GUI in a PSE for high-speed civil transport design. The interface allows users to provide and modify values of design variables. WBCSim [15,16] has a high-level GUI available through a Web browser that lets a user control the design variable values and shows the progress of the ongoing simulation.

Many PSEs involve large subsystems that were developed independently. The design of interface protocols for such components is an interesting research issue [2]. The model–view–controller (MVC) design pattern provides a clean separation between model (computational units) and view (user interface) [27]. In our work, the model consists of database and computational units, the controller comprises the database APIs exposed by an experiment manager, and the view is the GUI.

From the start, visualizations have been an integral part of PSEs. Some early PSEs were designed to only visualize data [28]. Traditionally, data analysis and visualization were performed as post-processing steps after a simulation has been run. As simulations have increased in size, the visualization task has become increasingly difficult, often requiring significant computation, high-performance machines, high-capacity storage, and high-bandwidth networks [29]. Messac and Chen [30] have developed PhysPro, a MATLAB-based application for visualizing an optimization process in real time using the physical programming paradigm. The Visual Computing Environment (VCE) provides coupling between flow analysis codes. The Multi-Disciplinary Computing Environment (MDICE) [31] provides users with a visual representation of the simulations being performed. In its distributed environment, many computer programs operate concurrently and cooperatively to solve a set of engineering problems. VizCraft [21] uses parallel coordinates to visualize the design metrics for high-speed civil transport design within a PSE. WBCSim [15,16] integrates visualization tools in a PSE for simulating wood-based composites. Oscill8 [32] is a system for bifurcation analysis of complex dynamic systems.

Interactive visualization can be defined [12] as viewing the results of simulations during or after the simulation run, with the user having control over the content and format of the graphics during their production. A further extension of this concept is *computational steering* [33], where computational resources are 'steered' towards a specific interesting process during the simulation [12]. Steering refers to modifying the simulation itself during the course of the simulation. GRASPARC [28] integrated computation and visualizations so that the scientist could view a computation as it proceeded. Moreover, a data recording facility was incorporated to help scientists backtrack to earlier points of a computation and restart. GRASPARC continued with the KINEX project, integrating CHEMKIN and IRIS Explorer [34] to create special-purpose environments for computational chemists. EPSRC DIVA integrated design, analysis, and visualization into a coherent environment based on IRIS Explorer [35].

Commercial visualization tools such as modular visualization environments have been tried in PSEs [36]. However, generic visualization tools are seldom useful in PSEs, since the problem domain dictates the visualization requirements. In most PSEs, *ad hoc* visualization tools are developed along with the modeling code [6]. In our work, we have developed a visualization tool for each of the simulation components to serve the specific needs of S^{4}W users.

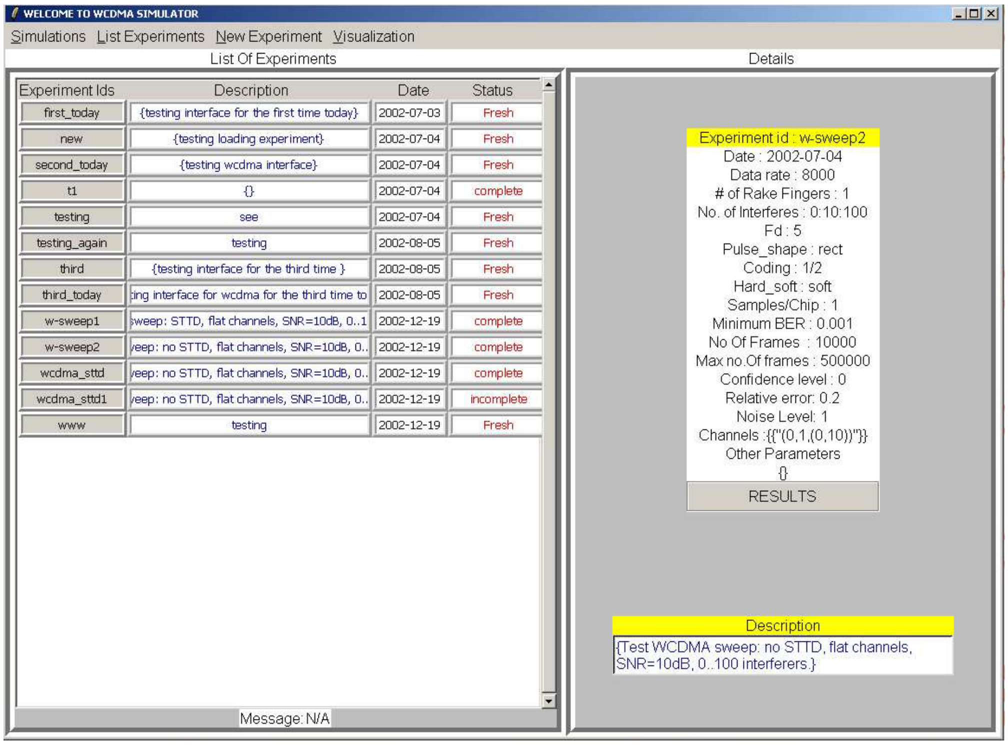
3. THE S^{4}W GUI

S^{4}W provides a unified user interface for its computational and data components. It provides users with the capability of starting a simulation, monitoring the run of a simulation with appropriate feedback, and viewing the results as they become available. The GUI is designed to retrieve simulation data (both input parameter values and output results) and display them in an appropriate form. The GUI is written as a Tcl/Tk script, which connects to a database using the Tcl database connectivity library `pgtcl` [37], and uses database functions to control the simulation runs.

The GUI also provides a facility to retrieve results from the database and visualize them. The visualizations themselves are described in Section 4. The GUI for S^{4}W consists of three parts: (i) the WCDMA simulator GUI, (ii) the ray tracing simulator GUI, and (iii) the DIRECT optimizer GUI. All three tools are available from the drop-down menu on the S^{4}W initial screen.

3.1. WCDMA simulator interface

The WCDMA interface was built on the experiment manager EMDAG [38] and provides a graphical view of EMDAG's text command lines. The WCDMA simulation itself runs on a 180 node parallel



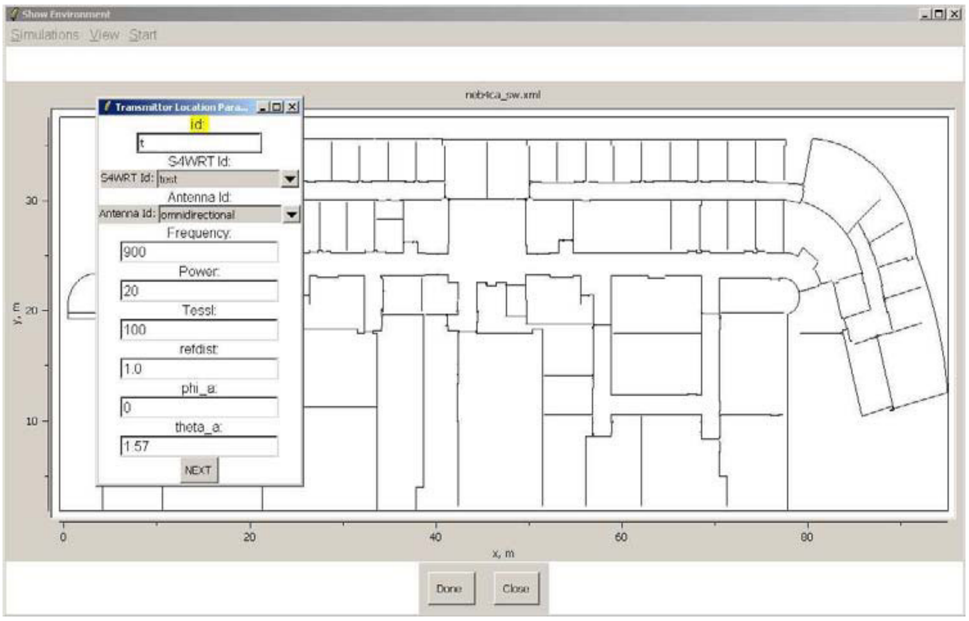
Experiment Ids	Description	Date	Status
first_today	{testing interface for the first time today}	2002-07-03	Fresh
new	{testing loading experiment}	2002-07-04	Fresh
second_today	{testing wcdma interface}	2002-07-04	Fresh
t1	{}	2002-07-04	complete
testing	see	2002-07-04	Fresh
testing_again	testing	2002-08-05	Fresh
third	{testing interface for the third time }	2002-08-05	Fresh
third_today	:ing interface for wcdma for the third time to	2002-08-05	Fresh
w-sweep1	sweep: STTD, flat channels, SNR=10dB, 0..1	2002-12-19	complete
w-sweep2	veep: no STTD, flat channels, SNR=10dB, 0..	2002-12-19	complete
wcdma_sttd	veep: no STTD, flat channels, SNR=10dB, 0..	2002-12-19	complete
wcdma_sttd1	veep: no STTD, flat channels, SNR=10dB, 0..	2002-12-19	incomplete
www	testing	2002-12-19	Fresh

Experiment id : w-sweep2
Date : 2002-07-04
Data rate : 8000
# of Rake Fingers : 1
No. of Interferes : 0:10:100
Fd : 5
Pulse_shape : rect
Coding : 1/2
Hard_soft : soft
Samples/Chip : 1
Minimum BER : 0.001
No Of Frames : 10000
Max no.Of frames : 500000
Confidence level : 0
Relative error: 0.2
Noise Level: 1
Channels :{{"(0,1,(0,10))"}}
Other Parameters
{}

Description
{Test WCDMA sweep: no STTD, flat channels, SNR=10dB, 0..100 interferers.}

Figure 2. A listing of experiments for WCDMA simulations. In the right pane, we have details for one experiment. At the bottom of the pane, there is another text box showing a small description of the experiment.

cluster with the database on a machine outside the cluster and a user interface on any other machine that supports the Tcl/Tk interpreter. The WCDMA GUI is implemented in the following three parts. (i) **List Experiments** retrieves WCDMA simulation results stored in the database with their experiment ID, a user's description of the experiment, the starting date of the experiment, and the status of the simulation (Fresh, Running, Complete, or Crashed; see Figure 2). Users can start a new experiment from here, or stop a running experiment. (ii) **New Experiment** opens a form to be filled in for a new simulation run. From here we can store the parameters in the appropriate tables of the database and invoke the database function to start the simulation on the computing unit. Storing a set of input values can be useful in order to define a template for creating variations on experiments. Using the **clone** command from the **List Experiments** view, we can reuse the same input parameter set and start an actual run later on. (iii) **Visualization** gives options for various visualizations available for stored results.



id: t
S4WRT Id: test
Antenna Id: omnidirectional
Frequency: 900
Power: 20
Tessl: 100
refdist: 1.0
phi_a: 0
theta_a: 1.57

Figure 3. View of the environment map, where devices (transmitters and receivers) are to be placed. Clicking on a transmitter causes the form on the left-hand side to pop up, which allows the user to set parameter values for that transmitter.

3.2. Ray tracer interface

Given a three-dimensional site description and transmitter locations, the ray tracer can approximate impulse responses at chosen receiver locations. A full ray tracing run can be divided into two parts: environment preprocessing, and parallel 3D ray tracing. Once an environment is preprocessed, it can be reused in many ray tracer runs. Preprocessing involves (a) converting floor plans from Autocad's DXF data format to the XML format understood by S^{4}W, (b) triangulating the polygons (walls) in the floor plan, and (c) partitioning the triangles within an octree representation to improve the speed of the ray tracing. While these three options are sequentially dependent, the interface presents them independently so that the steps can be performed at different times, since the output of each of these is stored in the database and can be used later on as input for the next step. We will refer to the resulting building description as an *environment file*. The ray tracer requires as input the device locations, device parameters, and propagation simulation parameters.

Selecting an environment file draws the environment map on the screen (see Figure 3). The location of transmitters is set on this screen by the user. A form to set the device parameters for a transmitter is also shown in Figure 3.

3.3. Optimizer interface

We now describe the GUI for VTDIRECT, a numerical optimizer based on the DIRECT algorithm of Jones *et al.* [5]. One goal of S^{4}W is to support transmitter placement optimization. We wish to achieve a certain performance objective, given the cost constraints [39]. The optimization process uses the full system, as it involves 3D ray tracing runs to predict impulse responses at points of interest within the environment for the transmitter locations sampled by the optimization algorithm. The ray tracer estimates power coverage levels, which are used as the criterion for optimization. As an alternative optimization criterion, Monte Carlo WCDMA simulations are used to estimate the bit error rates (BERs). In practice, surrogate functions for the WCDMA simulator are typically used to estimate the BERs, as the full WCDMA simulation is computationally intensive. We use both simple least-squares fits and multivariate adaptive regression splines [40] as surrogate functions for the WCDMA simulation.

The GUI for the optimizer consists of three parts, **Preview**, **New**, and **Visualization**. Environment files already stored in the database can be used for new experiments. The **preview** tab gives a preview of all the stored environment files and draws selected files on screen. This utility is useful when there are many environment files stored and we need to pick the right one to plan an experiment.

An optimization run is an iterative process that invokes one or more runs of the ray tracer to compute the impulse responses. These in turn are used to evaluate objective functions. Then new transmitter locations are constructed according to the optimization algorithm. This process stops when one of the optimization stopping criteria is met. An optimization experiment consists of (a) the ‘slave’ ray tracing experiment used for objective function evaluation, (b) one or two objective functions, (c) two or more optimization variables, and (d) additional optimization parameters.

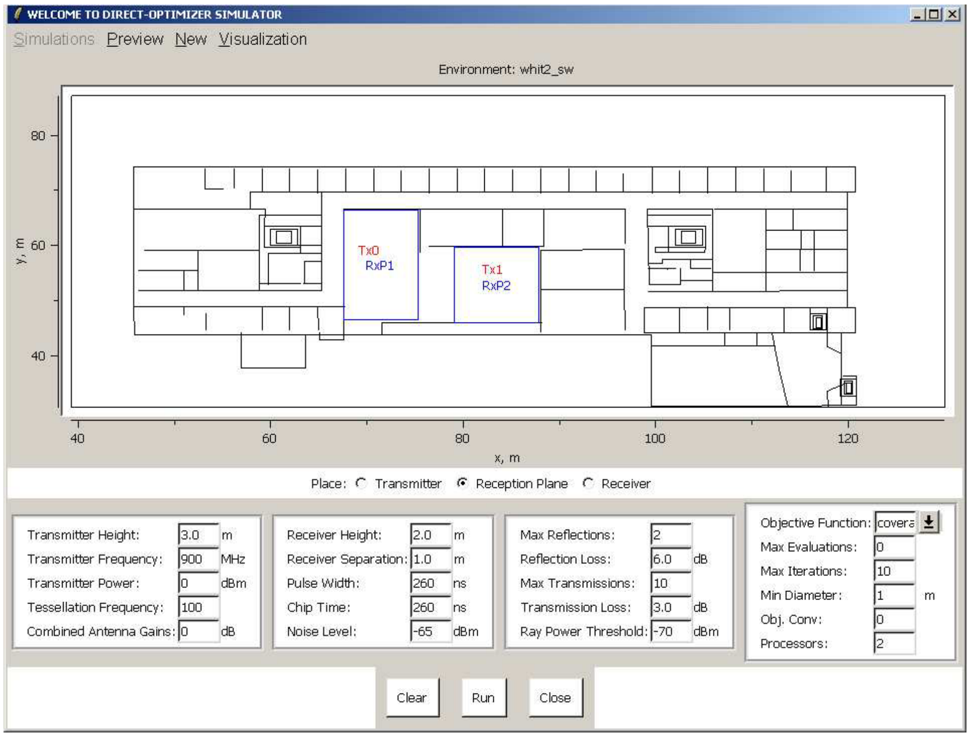
We need to specify a ray tracing simulation template to be used by the optimizer in successive iterations to create new transmitter location points. Defining a template is the same as defining a stand-alone ray tracing experiment. It stores the various parameter values for devices, which are later used by the optimizer for invoking the required ray tracing runs.

Once the ray tracer template is determined, we can set the optimizer parameters. The optimization process can be viewed as a master controller with a loop that calls the ray tracing simulation. A GUI is shown in Figure 4. Objective functions can be chosen from the drop-down list toward the bottom-right corner.

Initial transmitter placement in the environment can be done using a left-click after choosing the **transmitter** radio button. We can enclose each transmitter within a box, which specifies the area under which the transmitter can be moved by the optimizer. When choosing a transmitter, we create a pair of optimization variables: the x and y coordinates of the transmitter. When the optimizer is started, a dialogue box displays the debug messages generated by the simulation run. As soon as an iteration is complete and we obtain a value for the objective function a pop-up window shows a plot of the objective value against the number of iterations so far. This gives feedback for the ongoing simulation. Intermediate feedback is helpful since optimization runs typically last for hours.

4. VISUALIZATIONS FOR S^{4}W

Visualization can facilitate data analysis by providing a better insight into the problem, thus allowing novel interpretations of simulation data. In S^{4}W, visualization allows the designer to investigate


WELCOME TO DIRECT-OPTIMIZER SIMULATOR
Simulations Preview New Visualization
Environment: whit2_sw
Tx0
RxP1
Tx1
RxP2
y, m
x, m
40
60
80
100
120
Clear
Run
Close

Place: ( ) Transmitter [✓ Reception Plane] ( ) Receiver

Transmitter Height:	3.0	m
Transmitter Frequency:	900	MHz
Transmitter Power:	0	dBm
Tessellation Frequency:	100	
Combined Antenna Gains:	0	dB

Receiver Height:	2.0	m
Receiver Separation:	1.0	m
Pulse Width:	260	ns
Chip Time:	260	ns
Noise Level:	-65	dBm

Max Reflections:	2	
Reflection Loss:	6.0	dB
Max Transmissions:	10	
Transmission Loss:	3.0	dB
Ray Power Threshold:	-70	dBm

Objective Function:	covera	
Max Evaluations:	0	
Max Iterations:	10	
Min Diameter:	1	m
Obj. Conv:	0	
Processors:	2	

Figure 4. View of the GUI for the optimizer with two transmitters. Each of the transmitters is enclosed inside a box, which determines the bounds on transmitter placement. The entry boxes toward the bottom of the image take parameter values for optimizer runs.

alternative designs and formulate a new experiment based on past simulation results. S^{4}W provides separate visualizations for each of the three components of the system. The database acts as the backbone of the visualization system. Upon a request to visualize simulation data, the front end of the system connects to the database and retrieves results in a form suitable for visualization.

A series of WCDMA simulation runs are often made in order to study the effects on BER and frame error rate (FER) caused by changes to a particular parameter value. Typically, the collection of runs involves one parameter 'stepping' or changing value by a constant amount at each iteration. An additional measure of performance is the number of frames simulated. The WCDMA visualization provides a plot of the varying parameters against one of the output values (BER or FER). For each experiment, we can choose one of the input parameters and one of the output parameters and view the result in plot form. A tabular form is also provided so that users can see the values of other parameters along with the results. Users can zoom in or zoom out of a plot.

Sometimes a designer might be interested in comparing the results of more than one experiment with similar sweeps over one parameter and changing values for some other parameters. The **Visualization** button lets users compare the results from any number of experiments. They can filter the results retrieved from the database and choose the experiments by keyword in the ID or description. Users can choose any number of experiments using the check buttons to visualize the experiments collectively. Once the experiments are selected, users choose between a tabular or a graphical display. Users have all the display functionalities available for one experiment, with the only difference being that now we have simultaneous results for multiple experiments. The graph view shows the iteration points for different experiments with different colors. For the tabular results, a union of results from database queries for different experiments is shown.

A ray tracing run computes impulse responses in the form of a power delay profile (PDP) at specified locations. A PDP is a measure of the power received from the impulse at certain time/distance intervals. The ray tracer records the ‘raw’ impulse responses generated by the computation. One impulse response is recorded per pair of transmitter location and receiver location antennas. The composite impulse response consists of a number of rays. Each ray is characterized by an arrival time, a power, a phase, and arrival and departure angles [38]. A PDP is the only output of the ray tracer and is stored as a special data type.

The complex nature of the ray tracer output demands a sophisticated visualization tool. This output requires many steps of post-processing before the ray descriptions are converted to a standard PDP format. For this post processing, we can extract the information in the form of a value for the peak power of each arriving ray and then visualize this form of PDP. In this implementation, we create an XML file containing the time of arrival and peak power for each of the arriving rays. This file is then fed to the visualization tool, which draws it on the graph.

A run of the ray tracer starts by placing a transmitter in the environment and then measuring impulse responses at receiver locations. The ray tracer visualization draws the environment map on the screen and shows transmitter location and receiver locations (see Figure 5). Receiver locations can either be a point or an area that is divided into a grid, and impulse responses are measured at grid points. When a receiver location is clicked, a window pops up and shows the PDP at that point.

The ray tracer has been validated and calibrated with a series of measurements [38,41]. For this purpose, the impulse response was measured at seven different locations for a transmitter placed in the corridor of the fourth floor of Durham Hall, Virginia Tech (see Figure 5). Both measured and predicted values of PDPs at these locations are available. A visualization of the comparison between measured and predicted PDPs reveals the level of similarity between the two and validates the implementation of the ray tracer. Figure 5 shows the transmitter and receiver locations. A click on any of the receiver locations causes a window to pop up with the PDPs for both measurement and prediction (see Figure 6). By clicking on a transmitter’s location, we can see all the PDPs (at all receiver locations) in the same window. A zooming facility is available that allows the user to select any desired part of the graph, as shown in Figure 6.

The optimizer predicts the positions of transmitters for the optimal performance of the wireless system for a chosen region. We start the experiment by selecting an area of interest on the environment map. We then place transmitters with their bounds defined within the area of interest.

Our GUI provides visualizations showing the results of the optimization. At the end of an experiment, we obtain a final placement for the transmitters. Corresponding to this placement, we obtain the final value of power levels inside the area of interest. A color map showing the intensity of the power coverage gives design engineers a feel for the quality of the optimal coverage.

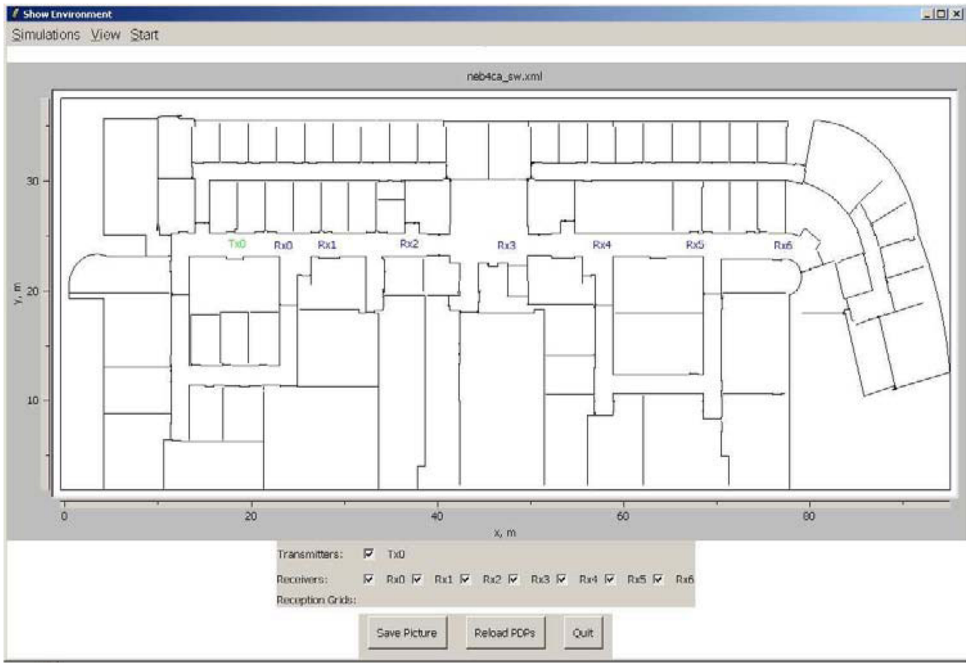


Figure 5. An environment map showing transmitter (Tx) and receiver (Rx) placements.

Under the **Visualization** tab of the Optimizer GUI we can retrieve all stored optimization experiments. The filtering of experiments based on keywords is also available, i.e. we can retrieve a subset of all experiments whose experiment ID or description contains some keyword. With a left-click on **Experiment ID**, a box containing details of the experiment and a smaller box below it with a small description of the experiment appears on the right pane, in a layout similar to that of Figure 7. To visualize the results of the experiment, either we click on the **Results** button at the bottom of the **Details** box, or select **Visualize** from the pop-up menu that comes with a right-click on **Experiment ID**. Two options are available for visualization: transmitter location and power coverage visualization, and objective function value visualization.

Selecting **Transmitter location visualization** brings up a window showing the environment map. The area of interest is filled with color values indicating the power coverage for the optimal transmitter placement (see Figure 8).

5. COMPARISONS

Sections 3 and 4 describe how S^{4}W is now an integrated system with a high-level graphical user interface and *ad hoc* visualizations. S^{4}W is intended to provide convenient user interaction

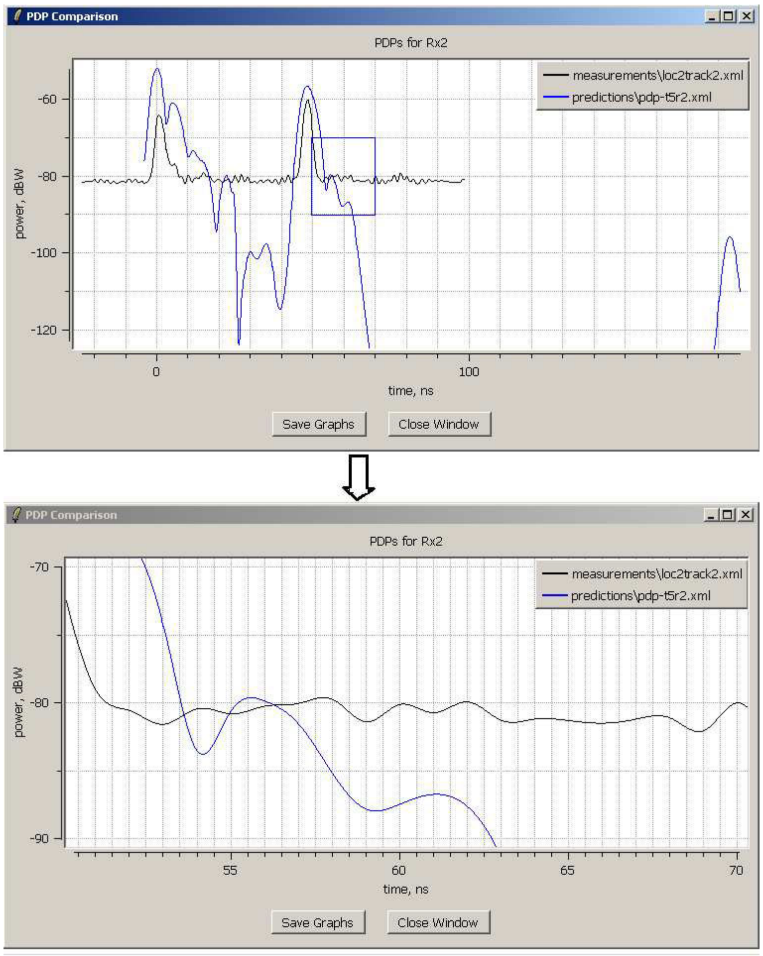
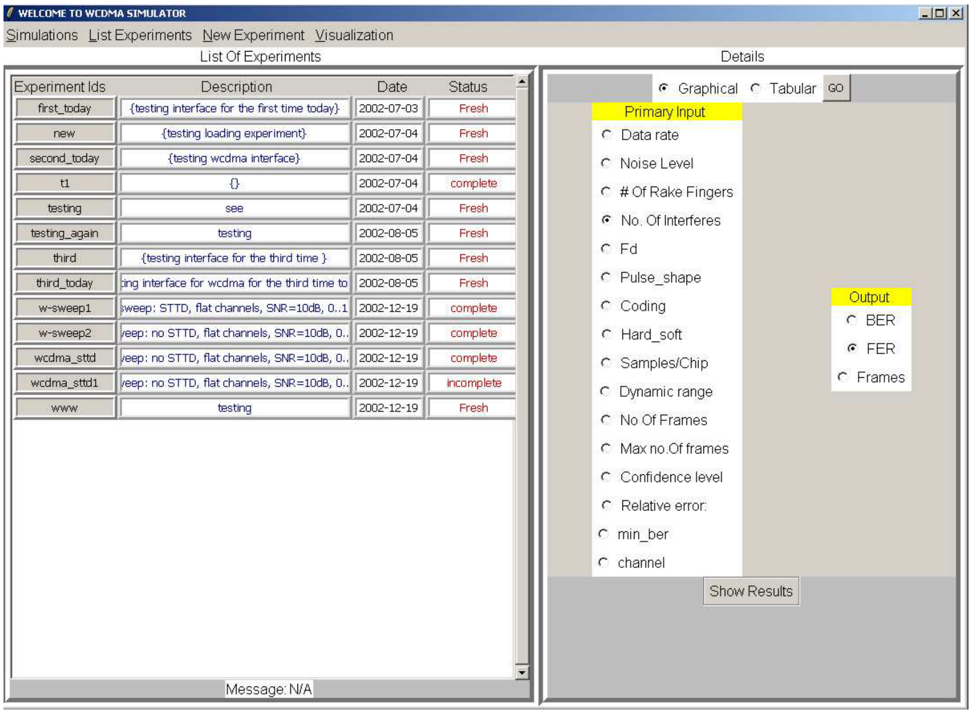


Figure 6. Simultaneous visualization of the PDPs obtained by measurement and a run of the ray tracer. The figure at the top shows the original view, with an area to be zoomed in shown in the rectangle. The figure at the bottom shows the selected area zoomed in.

by encapsulating the underlying implementation and connectivity details. To verify the claim, we compared S^4W as it existed before this integration and visualization work against the integrated version of the system.

Our comparison is based on the following use-case scenario for S^4W [3]. We assume that a wireless design engineer wishes to study transmitter placement in an indoor environment located on the fourth floor of Durham Hall at Virginia Tech. The engineering goal is to achieve a certain performance objective, given the cost constraints. For a narrowband system, power levels at the receiver locations are


WELCOME TO WCDMA SIMULATOR
Simulations List Experiments New Experiment Visualization
List Of Experiments
Details
GO
Show Results

Experiment Ids	Description	Date	Status
first_today	{testing interface for the first time today}	2002-07-03	Fresh
new	{testing loading experiment}	2002-07-04	Fresh
second_today	{testing wcdma interface}	2002-07-04	Fresh
t1	{}	2002-07-04	complete
testing	see	2002-07-04	Fresh
testing_again	testing	2002-08-05	Fresh
third	{testing interface for the third time }	2002-08-05	Fresh
third_today	:ing interface for wcdma for the third time to	2002-08-05	Fresh
w-sweep1	;weep: STTD, flat channels, SNR=10dB, 0..1	2002-12-19	complete
w-sweep2	/eep: no STTD, flat channels, SNR=10dB, 0..	2002-12-19	complete
wcdma_sttd	/eep: no STTD, flat channels, SNR=10dB, 0..	2002-12-19	complete
wcdma_sttd1	/eep: no STTD, flat channels, SNR=10dB, 0..	2002-12-19	incomplete
www	testing	2002-12-19	Fresh

Message: N/A

[✓ Graphical] Tabular

Primary Input
- Data rate
- Noise Level
- # Of Rake Fingers
- [✓ No. Of Interferes]
- Fd
- Pulse_shape
- Coding
- Hard_soft
- Samples/Chip
- Dynamic range
- No Of Frames
- Max no.Of frames
- Confidence level
- Relative error:
- min_ber
- channel

Output
- BER
- [✓ FER]
- Frames

Figure 7. The listing of experiments for WCDMA simulations. In the right pane we have input parameters in the left column and output parameters in the right column. Here we can choose one input parameter and one output parameter.

good indicators of system performance. Thus, minimizing the average shortfall of power with respect to some threshold power value can be a good objective. Here, the major cost constraints are the number of transmitters and their powers. Different placements of transmitters yield different levels of coverage. Our goal is to find the optimal placement of transmitters.

We repeated the original published study (which used an unintegrated collection of tools) with the integrated S^{4}W system. We achieved the same results in significantly less time. The original results could also be presented in a better way using the new visualization tools. This experiment was originally performed using console-based commands and with no visual feedback. As the first step, the user created a template of a ray tracing experiment, to be used by the optimizer as the starting point for its first iteration. This involved setting various parameter values for the wireless devices. The next step was to specify various parameter values for the numerical optimizer. Finally, the simulation was started. This whole process involved a series of database operations of insertions and selections. In the original version, the user was required to create a list of SQL commands to store the various parameters for the experiment in the order specified by the schema of the corresponding database tables (for details see [38]). That process required a high level of knowledge of the database schema and also some idea

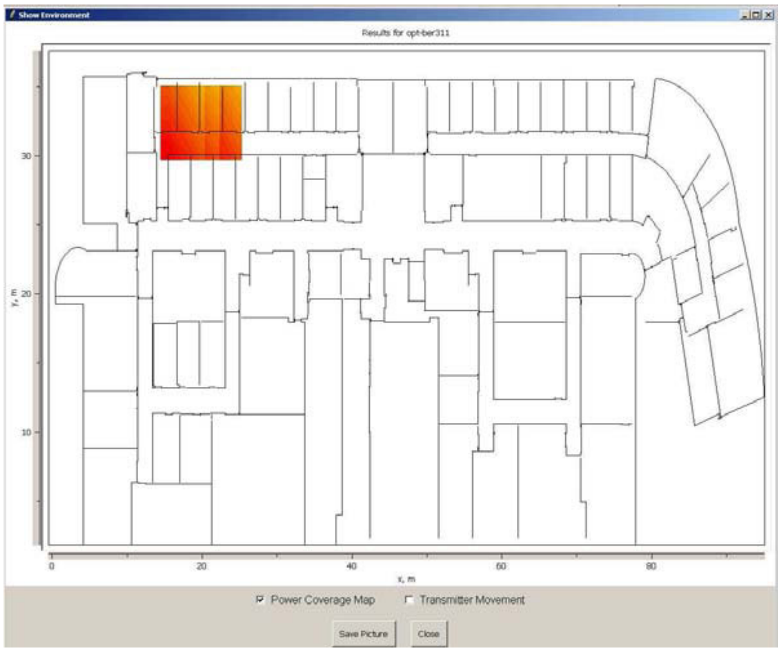


Figure 8. Visualization of power coverage on the environment map. Intensity of the color is proportional to the power level.

of how a database works. These requirements are the antithesis of a PSE, where the system is to be used by domain experts (electrical engineers), not system tool experts. In the integrated S^4W, the GUI takes care of all these issues.

The advantages of the current system over the old one are as follows:

1. it has relieved users of the burden of knowing about the database schemas of tables and their interrelations;
2. users do not need to know about database functions exposed by the experiment manager;
3. users do not need to worry about errors in the SQL commands, as error checks are incorporated into the GUI;
4. the GUI package is portable and works on any machine connected to the network.

We have tried to develop appropriate metrics to quantify the amount of effort put in by users of the two versions of the system. Users interact with a GUI with mouse clicks. Mouse clicks can be thought of as the basic operation in interacting with a GUI, while for a console-based user interface the basic operation can be thought of as pressing a key. However, these two operations cannot be directly compared. A more conservative metric is the number of lines for the console-based user interface. A command line in the original interface is presumably replaced by some series of mouse clicks in the GUI. This might be a difficult judgement to determine the relative 'cost' of a command line versus

Table I. A comparison of performance between a console-based user interface and the GUI for S^{4}W. The middle two columns show the average time required for the listed tasks. The last two columns show the number of mouse clicks required in the GUI versus the number of command line types typed in the console interface.

Task	Time taken with GUI (min)	Time taken with console user interface (min)	Number of mouse clicks	Number of lines on console
Setting a run of optimizer	5–10	30–45	20–35	95–120
Setting a run of ray tracer	5–10	25–40	18–30	70–90
Setting a run of WCDMA	1–5	20–35	6–20	40–75

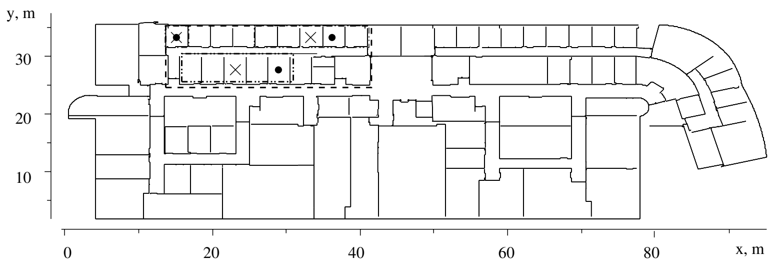


Figure 9. A visualization of transmitter placement from an older version of S^{4}W. A separate program generated the environment map and the rest of the figure was drawn using generic tools. The initial and final placements for transmitters (large black dots and crosses, respectively) are shown.

one or more mouse clicks. Surprisingly, we found that the number of mouse clicks for the integrated GUI is actually far less than the number of command lines originally required!

A more 'natural' metric we could use is the average time spent creating an experiment and sending it to the computational unit for each implementation. A major concern with using the time to develop the experiment as a metric is that the times might be heavily influenced by the level of user expertise. The command line interface is likely to be especially sensitive to this concern. It is worth noting that the command line versus mouse click comparison is more objective in this respect, even if it seems less natural. In our comparison, both versions were performed by an expert user, which should minimize this effect.

We measured the two user interfaces with respect to these two metrics. The results are given in Table I. We see that the performance of the integrated system is far superior to that of the unintegrated system. Thus, the GUI and integrated visualization tools appears to have a major effect on user performance.

To visualize the final position(s) of the transmitter(s), an *ad hoc* script had been written for the original experiment. This script drew the environment map, and the final locations of transmitters were marked on the map manually, as shown in Figure 9. Other forms of visualizations used in [3] were generated using generic graphics tools. With the integration of visualization tools for all three components, S^{4}W can now generate sophisticated visualizations on its own.

6. CONCLUSIONS

The S^{4}W project provides many experiences to share. It successfully integrates a GUI, including an interactive visualization tool, with back-end computation (by means of a scripting language), integrates data management and execution management by means of a relational database, incorporates the DIRECT global search algorithm, and replaces expensive computation by computationally cheap surrogate functions. We can outline the main contributions of S^{4}W to the PSE community and take stock of lessons learned from our experience as follows.

One of the primary contributions of this research is integration of data management and simulation management by means of a relational database. The S^{4}W architecture uses a Postgresql database as its hub. User interface and computing units interconnect to the database, and most data processing is performed via the database using stored functions written in C and PLSQL. This topology is suitable for data-centric simulation experiments. It facilitates data analysis and optimizes data movement across the network, and separates the computing platform from the user interface. We can direct computation to any platform according to our needs. Most of the intensive computation was performed on a parallel cluster, but small simulations (computationally not so extensive) were run on single processors.

For heterogeneous systems, scripting has often been the glue used to create an integrated system. For our system, we had the option of using Java or scripting. We selected Tcl/Tk as the scripting language for integration and it turned out to be a good choice. In most PSE implementation, groups of people from different backgrounds work together. Development often runs in parallel with different choices of programming platforms and programming languages. The final stage is integration, where we need to take the pieces and glue them together to create the integrated system. For S^{4}W we had such a situation. We had different components written in different languages such as C, FORTRAN, and MATLAB, and on different platforms such as Sun Solaris and Microsoft Windows. Our job was to create a portable user interface. Tcl/Tk suited all our needs and worked as the glue for all of the components. A user interface written in Tcl/Tk is platform independent. The S^{4}W GUI is now portable to many platforms, and has been tested on Microsoft Windows and Sun Solaris platforms. We could have achieved this goal using Java, but Tcl/Tk scored better with extensive string manipulation capability, excellent network connectivity, and good support for database connectivity. Our visualization tools provide the user with the capability to view the stored simulation results in various domain-specific formats, and also to compare results of different simulations. Since our visualizations were written using Tcl/Tk, they integrated easily with the rest of the system.

Overall, we had a good experience using Tcl/TK as a medium for the integration of heterogeneous PSE components and to provide a suitable GUI. Since we were developing a proof-of-concept prototype, the flexibility that the higher-level language Tcl/TK provided for gluing together the disparate parts was an important factor in our ability to perform rapid prototyping. We went through many iterations of our user interfaces, which is typical when designing novel components for a PSE. A more mature software development effort, such as a reimplementation of the system as a production facility, might benefit from the more mature development environment that Java now provides. As a rapid prototyping tool, Tcl/Tk proved of value to our efforts.

Even the fastest computers or a collection of such computers may not be good enough for some computations. For Monte Carlo simulations of a channel model, it takes days to run the simulation on a 180 node parallel cluster for a few hundred points. To reduce this cost while maintaining adequate fidelity for the behavior of the WCDMA simulation, we used a multivariate adaptive regression spline as a surrogate model. The surrogate function allowed us to effectively simulate the behavior of the original function with six orders of magnitude less computational burden.

A typical PSE consists of three phases: data (simulation input) preparation, simulation (or measurement), and data (simulation output) analysis. These three phases can be repeated and interleaved in arbitrary ways. In our approach, simulation corresponds to loading data into the database, data analysis corresponds to querying the database and presenting data visualization at the GUI level, and data preparation corresponds to a combination of loading and querying. Objective function evaluation for the optimizer was implemented using simple SQL queries, and format conversion and surrogate models for PSE data are implemented as (manually) stored database functions. Thus, we have shown that appropriate use of a state-of-the-art database management system can provide high-level problem-solving facilities.

A sufficiently flexible relational database can act as interpreter for a scripting language specifically designed for data processing. A library of custom data types, functions, and aggregation operators naturally specializes this language for a particular application domain. This overloading of query semantics within the application domain has many potential uses. Not only does this approach simplify format conversion of the simulation data, it can ultimately change the way users look at their experiments. Parts of simulations can, in principle, be implemented as database queries [38].

The database approach to experiment management has been explored in other projects. The ZOO desktop experiment management environment [42] aims to achieve goals similar to ours, but lacks some necessary features such as support for parallel simulation. Virtual Cell [13] also has the database as its central component. We adopted a hybrid approach that incorporates both simulation data management and simulation execution management to create the experiment manager EMDAG for S^{4}W [38]. This experimentation demonstrated that data management can be usefully harnessed, in multiple ways, in a PSE.

Processes that are well defined, and intellectually simple, can be captured in terms of a specific workflow through the system. In those situations, the system can give the user strong guidance and support through the workflow, as is the case of 'wizard' systems such as tax preparation software. However, solving difficult problems is very much an exploratory process. As such, users need the freedom to investigate and try out things. This typically means that they take an unpredictable path through the various tools provided by the PSE. While relational schemas enforce constraints on workflow, they enforce no centralized workflow management. This provides a good compromise between supporting the user through specialized tasks while allowing flexibility to switch between exploratory tasks at will.

Decision-making processes in PSEs are often complex and require support by automated optimization techniques. S^{4}W uses the DIRECT optimization algorithm to find the optimal locations of transmitters. This integration turned out to be a success and this global optimization algorithm has proved to be an effective approach in solving transmitter placement problems [3].

This brings up the issue of a fundamental tension between the human exploration of a design space in what is actually an optimization problem versus limiting the human role to specifying the problem and allowing the automated optimizer to find the optimal solution. Ultimately, the goal of a PSE is to automate what can be automated, and allow the user to creatively solve what cannot be solved by the computer. Whenever an optimizer can reliably and efficiently solve the problem, it should be allowed to do so. However, for truly complex problems, even if the optimizer can find the optimal solution, the user will still need good visualization and some ability to 'twiddle' so as to be able to properly pose the problem. Within our context, this means that the user needs to be able to define the physical environment, and to specify constraints on transmitter and receiver locations. Not only does setting these constraints limit the scope of the optimization problem so as to make the computation practical, but it also is an important part of making sure that the correct problem is being solved.

Finally, high-level domain-specific schema editing tools could reduce the burden on PSE developers. No amount of point and click gadgets can make up for flaws in the schema design with the central database. Designing appropriate schemas and integrating these schemas with those of the other system components is ultimately the responsibility of the PSE developer. The most problematic part of schema development is the precise definition of simulation inputs and outputs. Once the simulation is well defined, the data model follows.

ACKNOWLEDGEMENTS

This project was funded under NSF grants EIA-9974956, EIA-9984317 and EIA-0103660.

REFERENCES

1. Rice JR, Boisvert RF. From scientific software libraries to problem-solving environments. *IEEE Computational Science and Engineering* 1996; **3**(3):44–53.
2. Gallopoulos E, Houstis EN, Rice JR. Workshop on problem-solving environments: Findings and reccomendations. *ACM Computing Surveys* 1995; **27**(2):277–279.
3. Verstak A, He J, Watson LT, Ramakrishnan N, Shaffer CA, Rappaport TS, Anderson CR, Bae K, Jiang J, Tranter WH. S^4W: Globally optimized design of wireless communication systems. *Proceedings of the Next Generation Software Workshop at the 16th International Parallel and Distributed Processing Symposium (IPDPS'02)*, Fort Lauderdale, FL, April 2002. IEEE Press: New York, 2002.
4. Rappaport TS. *Wireless Communications: Principles and Practice*. Prentice-Hall: Englewood Cliffs, NJ, 1996.
5. Jones DR, Perttunen CD, Stuckman BE. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications* 1993; **79**(1):157–181.
6. Shaffer CA, Watson LT, Kafura DG, Ramakrishnan N. Features of problem-solving environments for computational science. *Proceedings of High Performance Computing Symposium at the Advanced Simulation Technology Conference*, Tentner A (ed.). Society for Modeling and Simulation International: San Diego, CA, 1999; 242–247.
7. Mishra D. Integration of graphical user interface and data visualization tools in a problem-solving environment for wireless system design. *Master's Thesis*, Department of Computer Science, Virginia Tech, April 2004.
8. Boisvert RF, Rice JR. *Solving Elliptic Problems Using ELLPACK*. Springer: New York, 1985.
9. Yuen DA, Garbow ZA, Erlebacher G. Remote data analysis, visualization and problem-solving environment (PSE) based on wavelets in the geosciences. *Visual Geosciences* 2003; **8**:83–92.
10. Markus S, Weerawarana S, Houstis EN, Rice JR. Scientific computing via the world wide web: The net PELLPACK PSE server. *IEEE Computational Science and Engineering* 1997; **4**(3):43–51.
11. Ramakrishnan N, Watson LT, Kafura DG, Ribbens CJ, Shaffer CA. Programming environments for multidisciplinary grid communities. *Concurrency and Computation: Practice and Experience* 2002; **14**:1241–1273.
12. Watson LT, Lohani VK, Kibler DF, Dymond RL, Ramakrishnan N, Shaffer CA. Integrated computing environments for watershed management. *Journal of Computing in Civil Engineering* 2002; **16**(4):259–268.
13. Moraru II, Schaff JC, Slepchenko BM, Loew LM. The virtual cell: An integrated modeling environment for experimental and computational cell biology. *Annuals of the New York Academy of Science* 2002; **971**:595–596.

14. Bramley R, Chiu K, Diwan S, Gannon D, Govindaraju M, Mukhi N, Temko B, Yochuri M. A component based services architecture for building distributed applications. *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing (HPDC'00)*, 1–4 August 2000. IEEE Computer Society Press: Washington, DC, 2000.
15. Goel A, Phanouriou C, Kamke FA, Ribbens CJ, Shaffer CA, Watson LT. WBCSim: A prototype problem-solving environment for wood-based composites simulation. *Engineering with Computers* 1999; **15**:198–210.
16. Shu J, Watson LT, Ramakrishnan N, Kamke FA, Zombori BG. An experiment management component for the WBCSim problem-solving environment. *Advances in Engineering Software* 2004; **35**:115–123.
17. Alscher RG, Chevone BI, Heath LS, Ramakrishnan N. Expresso—a PSE for bioinformatics: Finding answers with microarray technology. *Proceedings of the High Performance Computing Symposium at the Advanced Simulation Technologies Conference*. Society for Modeling and Simulation International: San Diego, CA, 2001; 64–69.
18. Gordon JC, Myers JB, Folta T, Shoja V, Heath LS, Onufriev A. H++: A server for estimating pkas and adding missing hydrogens to macromolecules. *Nucleic Acids Research* 2005; **1**(33):368–371.
19. Allen NA, Shaffer CA, Vass MT, Ramakrishnan N, Watson LT. Improving the development process for eukaryotic cell cycle models with a modeling support environment. *Simulation* 2003; **79**(12):674–688.
20. Drashansky TT, Houstis EN, Ramakrishnan N, Rice JR. Networked agents for scientific computing. *Communications of the ACM* 1999; **42**(3):48–54.
21. Goel A, Baker CA, Shaffer CA, Grossman B, Mason WH, Watson LT, Haftka RT. VizCraft: A problem-solving environment for aircraft configuration design. *IEEE/AIP Computing in Science and Engineering* 2001; **3**(1):56–66.
22. Bonomo JP, Dyksen WR, Rice JR. The ELLPACK performance evaluation system. *Technical Report CSD-TR-569*, Department of Computer Sciences, Purdue University, 1986.
23. Weston RP, Townsend JC, Edison TM, Gates RL. A distributed computing environment for multidisciplinary design. *Proceedings of the 5th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 1994, vol. AIAA-94-4372; 1091–1097.
24. Tong SS, Powell D, Goel S. Integration of artificial intelligence and numerical optimization techniques for the design of complex aerospace systems. *Proceedings of the Aerospace Design Conference*, Irvine, CA, February 1992, vol. AIAA-92-1189. Available at:
http://md1.csa.com/partners/viewrecord.php?requester=gs&collection=TRD&recid=200114003961MT&recid=A9233301AH&q=Tong+Powell+Goel+Integration+numerical+optimization&uid=788071451&setcookie=yes
25. Guisset P, Tzannetakis N. Numerical methods for modeling and optimization of noise emission applications. *Proceedings of the ASME Symposium in Acoustics and Noise Control Software at the ASME International Mechanical Engineering Congress and Exposition*, November 1997. Available at:
http://md1.csa.com/partners/viewrecord.php?requester=gs&collection=TRD&recid=200121019115MT&q=Guisset+Tzannetakis+Numerical+Methods+Modeling+Noise&uid=788071451&setcookie=yes
26. Eldred MS, Hart WE. Design and implementation of multilevel parallel optimization on the intel teraflops. *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, September 1998, vol. AIAA-98-4707; 44–54.
27. Olsen D Jr. *Developing User Interfaces*. Morgan Kaufmann: San Francisco, CA, 1998.
28. Brodlie K, Poon A, Wright H, Brankin L, Banecki G, Gay A. GRASPARC: A problem-solving environment integrating computation and visualization. *Proceedings of the 4th Conference on Visualization (VIS'93)*, San Jose, CA, 1993; 102–109.
29. Johnson C, Parker S, Hansen C, Kindlmann GL, Livnat Y. Interactive simulation and visualization. *IEEE Computer* 1999; **32**(12):59–61.
30. Messac A, Chen X. Visualizing the optimization process in real-time using physical programming. *Engineering Optimization Journal* 2000; **32**(5).
31. Kingsley G, Siegel JM Jr, Harrand VJ, Lawrence C, Luker JJ. Development of a multi-disciplinary computing environment (MDICE). *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, 2–4 September 1998, vol. AIAA-98-4738; 251–260.
32. Conrad ED, Tyson JJ. Oscill8: A dynamical systems and bifurcation theory toolset. *Foundations of Systems Biology in Engineering (FOSBE 2005)*, August 2005.
33. Parker SG, Johnson CR, Beazley D. Computational steering software systems and strategies. *IEEE Computational Science and Engineering* 1997; **4**(4):50–59.
34. Wright H, Brodlie KW, Brown MJ. The dataflow visualization pipeline as a problem-solving environment. *Virtual Environments and Scientific Visualization*. Springer: Berlin, 1996; 267–276.
35. Wood J, Wright H, Brodlie K. Collaborative visualization. *Proceedings of the 8th Conference on Visualization (VIS'97)*, Phoenix, AZ, 1997. IEEE Computer Society Press: Los Alamitos, CA, 1997; 253–260.
36. Wright H, Brodlie K, Wood J, Proctor J. Problem solving environments: Extending the role of visualization systems. *Euro-Par 2000 Parallel Processing*. Springer: Berlin, 2000; 1323–1331.
37. Ousterhout JK. *An Introduction to Tcl and Tk*. Addison-Wesley: Reading, MA, 1994.

38. Verstak AA. Data and computational modeling for scientific problem-solving environments. *Master's Thesis*, Department of Computer Science, Virginia Tech, July 2002.
39. He J. Global optimization of transmitter placement for indoor wireless communication systems. *Master's Thesis*, Department of Computer Science, Virginia Tech, August 2002.
40. Friedman JH. Multivariate adaptative regression splines. *Annals of Statistics* 1991; **19**(1):1–141.
41. Anderson CR. Design and implementation of an ultrabroadband millimiter-wavelength vector sliding correlator channel sounder and in-building multipath measurements at 2.5 and 60 GHz. *Master's Thesis*, Department of Electrical Engineering, Virginia Tech, May 2002.
42. Ailamaki A, Ioannidis Y, Livny M. Scientific workflow management by database management. *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM'98)*, July 1998. IEEE Computer Society Press: Washington, DC, 1998; 190–199.