1994

Science Pad: An Intelligent Electronic Notepad for Ubiquitous Scientific Computing

Anupam Joshi

Sanjiva Weerawarana

Tzvetan T. Drashansky

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

Report Number:
94-061

SCIENCE PAD: AN INTELLIGENT ELECTRONIC NOTEPAD FOR UBIQUITOUS SCIENTIFIC COMPUTING

Anupam Joshi
Sanjiva Weerawarana
Tzvetan T. Drashansky
Elias N. Houstis

CSD-TR-94-061
September 1994
(Revised 12/94)

SciencePad: An Intelligent Electronic Notepad for Ubiquitous Scientific Computing*

Anupam Joshi Sanjiva Weerawarana Tzvetan T. Drashansky
Elias N. Houstis
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398

email: {joshi,saw,ttd,enh}@cs.purdue.edu
Fax: +1-317-494-0739

December 8, 1994

Abstract

The National Information Infrastructure (NII) that will evolve in the 1990's and beyond will impact many institutions of life. These include the way we learn and do science, self–caring, access to civil/information infrastructure systems & services, and management & control of manufacturing processes. The future scenario for the NII assumes wireless networks used by walkstations realizing the dream of truly ubiquitous access to the information superhighway. Some of the current obstacles in building such a ubiquitous access system on mobile high–performance platforms include the user interface for these walkstations, the ability of sniffing information across heterogeneous geographically distributed information systems, the ability of processing sensoring data for monitoring and control, and the dynamic reconfigurability of computations between the mobile unit and the stationary servers. Our effort in the area of ubiquitous computing involves the design and implementation of intelligent models and techniques to address the aforementioned obstacles. In this paper we present the architecture of an intelligent electronic notepad, called *SciencePad*, to support a scientist in the research environment of the future.

*This work was supported in part by NSF grants CDA-9123592 and ASC 9404859.

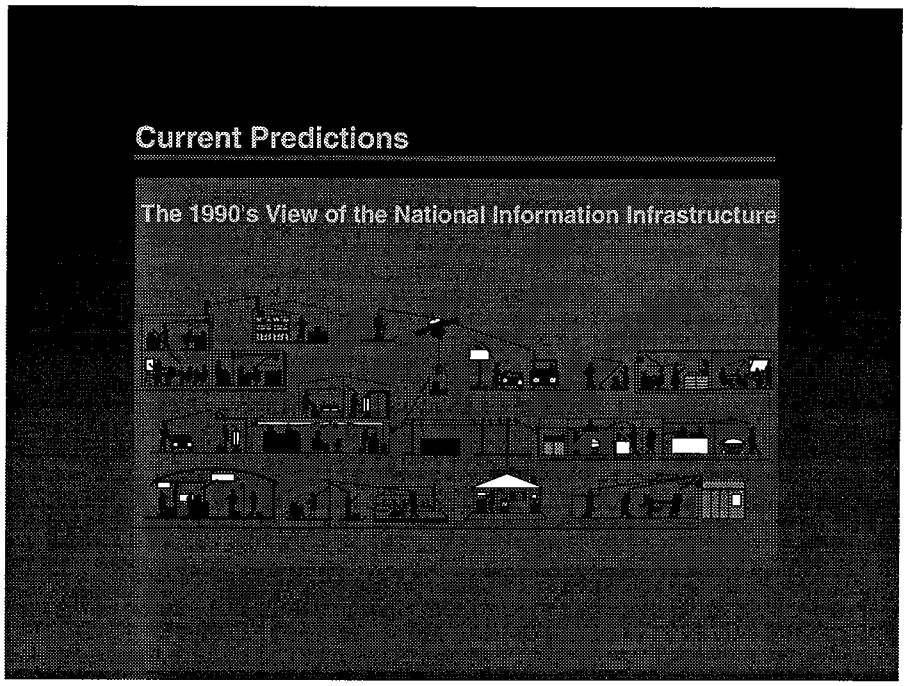

Figure 1: A Predicted Scenario for the Future National Information Infrastructure

1 Introduction

The National Information Infrastructure (NII) that will evolve in the *fin de siecle* and the early years of the next century has caught the public's imagination, and is popularly referred to as the information superhighway. The superhighway part of the NII will involve very high speed (Gbps and above) connections between major nodal regions. There will also be many state and regional highways. Yet these highways cannot truly provide ubiquitous access to the NII, the kind of access that country roads provide. The "country road" component of NII will consist of wireless networks used by *walkstations* - handheld mobile hosts - which will realize the dream of truly ubiquitous access. The NII will impact many aspects of social and academic life, it will engender a continuous interaction between people and interconnected computers using wireless devices. Figure 1 illustrates an impression of a predicted scenario for the NII.

This new paradigm, variously referred to as mobile, nomadic or ubiquitous computing, raises a host of research issues in a variety of areas of computer science [7]. The nature of the hardware involved in the mobile part of this scenario forces certain restrictions, like limited amounts of communication bandwidth, memory, and power. Another critical issue is the nature of the interfaces, the traditional WIMP (Window, Icon, Mouse, Pointer) type are unlikely to be successful. Still other issues arise because of the interaction and collaboration that such a paradigm foresees. The new trend in PC applications is their seamless integration within a common interface. Unfortunately, this cannot be done easily in scientific and engineering computing, since most of the available tools have been developed as stand alone systems. This issue must be seriously addressed in the context of mobile computing, since the physical limitations of the proposed devices determine the "natural" upper bounds on the facilities.

In this paper we present the design, architecture and implementation of an intelligent electronic notepad, called *SciencePad*, and its associated information management system.

2 The SciencePad Paradigm

In this section, we outline the new paradigm for scientific computing that we feel will be brought about by the advent of ubiquitous computing. The objective is to understand the needs of the scientist, and then to outline the facilities that we feel should be available to meet these needs.

One can currently identify three major tools in a scientists work today. Firstly, there is the traditional pen and paper. This is where most of the creative portion of a scientists

work gets done. One writes down ideas, does preliminary designs, plays around with symbolic equations, etc.. Then there is the (scientific) calculator. This is used for (relatively simple) numerical and symbolic computation. For more complicated numerical or symbolic calculations, the scientists uses another tool, the computer. The pen and paper have become ubiquitous tools over the centuries while the other tools mentioned above are still foci of attention. They distract the scientist from the thinking process and force him to divert attention to them in order to put them to proper use. It has been argued [8, 9], and correctly in our opinion, that the most efficient tools are those that are invisible, in other words tools that we use without thinking. The objective of *SciencePad* is to serve as such an invisible tool. In order to do this, *SciencePad* provides an interface which enables the scientist to call on the power of the sophisticated tools like scientific calculators and computers directly and seamlessly. To give an example, consider a scientist working on a design problem which requires an integral to be evaluated. Rather than forcing the the use of a calculator or computer, the interface of *SciencePad* recognizes when some symbolic/numeric expression is to be evaluated and it automatically accesses the needed resource, either locally or by contacting some remote computer.

This pen and paper kind of model of human computer interaction involves the design of completely new interface mechanisms [7, 9]. Such interfaces will of necessity be multimodal, as well as be endowed with innate intelligence.

In order to serve as an effective tool, we expect *SciencePad* to provide certain basic functionality. Clearly, in the context in which this tool is sought to be created, this would involve symbolic and numeric computing abilities. In addition, we expect *SciencePad* to be able to take pen and voice inputs. It should have the ability to understand basic shapes

and manipulate them. We would also want to have a miniature camera that would allow *SciencePad* to acquire images, and software that would allow it to do some amount of image processing. Besides these computation related capabilities, *SciencePad* should also be able to handle information. An important requirement for a scientist is the ability to access technical information. This information, though available, could be in any one of various distributed, heterogeneous databases. Also, the user may pose an extremely structured query (Jane Doe, "Some new developments in underwater basket weaving", *J. UWBW* Vol5, #1, pp 201-255), or an extremely unstructured one (Wasn't there an article on basketweaving by Jane Doe in the late 80's?). To address this issue, we exploit two existing systems. Much of the information available on the network today is linked together as the World Wide Web (WWW). It has associated with it a language in which to keep the information (HTML), and protocols on how to transfer it (HTTP, Gopher, ftp). *SciencePad* is equipped with a browser for such information. This will incorporate the ability to search the web for information pertaining to given parameters. To take into account the heterogeneous nature of the databases, we exploit *InterBase*, a technology being developed at Purdue [2].

These are the basic functionalities that we feel will allow a scientist sufficient power to do his work. The hardware to support these facilities is available today. What is missing is the software that provides users with integrated platforms to apply these facilities effectively. *SciencePad* aspires to fill this void.

2.1 Intelligence in Mobile Computing

In this section, we shall illustrate how AI helps in realizing the architecture proposed above, coping with the constraints imposed by mobile computing. Intelligence here refers to domain specific knowledge and information which would be available to the system. This knowledge is exploited by the various artificial / computational intelligence strategies to address the problems in designing a ubiquitous information management system. Intelligence is needed for the ability to take most decisions regarding the solution process automatically. That is, once a problem is specified by the user with some basic direction on how to solve it, *SciencePad* should be able to infer a solution strategy, as well as fill in any parameters the user has left unspecified. In the case of scientific computing, this involves, for instance, choosing the right numerical algorithm, the discretization parameters and the time steps, the machine on which to solve the problem and it's configuration *inter alia.* The ability of the system to make such decisions helps in making it more accessible to the non expert in the area, which is of extreme importance if scientific computing is to become truly ubiquitous (in the sense of access by persons who are not specialists in the area). Most current systems cannot be used as is by application scientists or students, but require users to develop some expertise in parallel and numerical programming. Further, in the ubiquitous scenario that we are considering for *SciencePad*, an important objective is to minimize the communication between the mobile device and the static hardware. Intelligence is needed to achieve this. For instance, the ability of the software intelligently infer parameter values saves much valued communication bandwidth. Traditional keyboard or mouse interfaces are not always available or useful for handheld devices; new pen–based, intelligent, multimodal interfaces are being designed. The user interacts with this inter-

face using a "natural like" language. This language is domain specific. To illustrate, the user should be able to say *solve modified Laplace equation, alpha = 2.1, identify region of maxima.* The system parses this input, generates the mathematical equation to be solved, chose an appropriate solution approach and returns the answer.

2.2 An Application Scenario

We now illustrate a scenario where *SciencePad* will be used. A researcher usually does some experiments in his/her lab, then evaluates the presumed theoretical model, or builds a new one, and schedules new experiments to refine the model. *SciencePad* attempts to aid the scientist in the thinking process, which is usually done using a notepad. A scientist could run an experiment for which s/he has a model involving Partial Differential Equations (PDEs) by controlling the equipments from *SciencePad.* The experiment can be monitored, and *SciencePad* can display the results of the experiment in some appropriate form necessary. Then the scientist can make some remarks about the model, change a parameter or just compare the run of the experiment with the results expected from the model. *SciencePad* can make sense of user remarks (in some constrained and domain specific language) on the basis of previous experience and suggest an action (say, solving PDE with the new parameters). Even if a task requires very detailed algorithmic description when it is done the first time, chances are that when it is done again, it will require at most minor changes in the parameters. *SciencePad* provides a framework where such previously defined tasks can be invoked from the scientist's notes. We believe that integrating the problem solving environment interface (in this case, the interface to a PDE solver) with the data from a knowledge base (for example, of results from similar experiments or of theoreti-

cal models for the studied event) into a notebook-like form will enhance the availability, the acceptability, and the usability of high performance computing facilities. The intelligent static host solving the PDE can take into account the context sent by the mobile host to display only the relevant parts of the computed results. Ubiquity by nature of space allows the computations to proceed at any place - in the front–end of the experimental machine, at a discussion table with other colleagues, at a conference far from the lab (for instance, checking how the results reported by the speaker match ours), etc.

After outlining a vision of *SciencePad* and a possible application scenario, we next describe the software architecture that we propose for such a system.

3 The Software Architecture of *SciencePad*

The software architecture of *SciencePad* is shown in Figure 2 and is based on clean layering. The lowest layer comprises the data and knowledge communication infrastructure for *SciencePad.* This layer contains a generic communication facility based on the software bus model and various knowledge communication facilities. *SciencePad* supports the ARPA Knowledge and Query Manipulation Language [3] and SciencePad Knowledge Interchange Format (SKIF), a *SciencePad*–specific knowledge communication format based on PDESpec [6].

The next level is the software infrastructure supporting user interfaces building and component cooperation. Software components are viewed as agents and this layer provides the infrastructure for building the agents needed for *SciencePad.* This layer is composed of an object manager supporting persistency for managing data objects, a notebook and

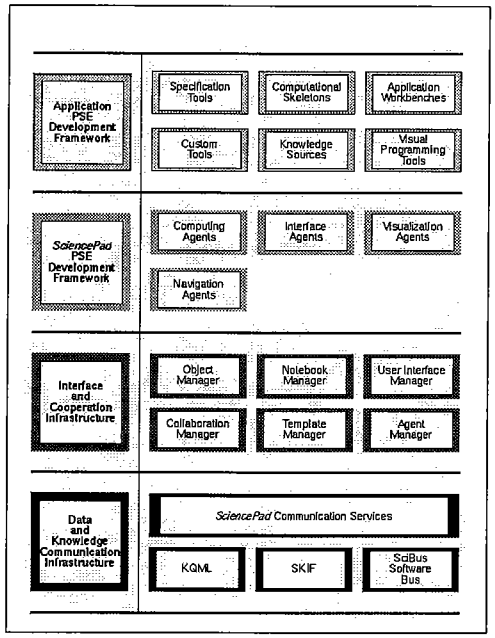

Figure 2: The *SciencePad* software architecture.

user interface manager for managing the overall working environment. The other managers (collaboration, template and agent) provide appropriate overall management functionality for their particular subsystems.

The development frameworks layer provides a collection of agents and other tools for building various intelligent information management tools. *SciencePad* is itself built using these agents and the services provided by the lower layers. The highest–layer represents specializations of *SciencePad* to domain–specific instantiations. For example, a *SciencePad* for nonlinear chromatography systems used for component separation in chemical engineering [1].

We expect that the hardware platform supporting *SciencePad* will not be able to execute most scientific computing applications. Hence, any such application is logically viewed as having two parts, a *proxy* and a *scion.* The proxy executes on the *SciencePad* platform and intelligently uses its scion to implement that application's functionality. This approach

allows us to make transparent the details of the location of and communication with any application to the interface systems which always interacts with locally executing tools.

4 Prototype Implementation

We have currently implemented a prototype of SciencePad. The mobile host is an NEC Versa notebook running Windows for Workgroups modified for pen support. The static backend runs on a variety of unix boxes, mostly Suns. The wireless link is a 2Mbps AirLan Ethernet on top of which we run TCP/IP. The domain of the intelligent information management system is chosen as partial differential equations (PDEs). The core application in the *scion* is //ELLPACK [4], a system to solve elliptic PDEs. *SciencePad* can be logically viewed as consisting of three subsystems, the notebook interface, the applications and supporting agents, and the means of communication between them. In the presents instance, the user interacts with *SciencePad* using the notebook. The users inputs are "semantically interpreted", by the application proxy, and communicated to its scion. The scion invokes various agents to fill in the sparse input provided by the user, and then invokes the applications. When the results are available, it sends them over to the proxy which uses the notebook to display them appropriately. We next describe these subsystems.

4.1 The Notebook Subsystem

The electronic notebook concept is an attempt to emulate the physical notebook that we use ubiquitously. It provides an unrestricted editing environment where users can record their problem and solution specifications, computed solutions, results of various analyses,

commentary text as well as handwritten comments. The notebook interface is multimodal and synergetic, it integrates text, handwriting, graphics, audio and video in its input and output modes. It functions not only as a central recording mechanism, it also acts as the access mechanism for all the tools that support the user's problem solving activities.

The current *SciencePad* notebook is implemented using NCSA Mosaic. It uses the underlying data and knowledge communication infrastructure, as well as the Interface and Cooperation infrastructure to invoke the tools requested by the user and is implemented in an application–independent manner via an indirect referencing scheme. Basically, every object that is visually or textually represented in the notebook is physically located elsewhere in the object manager. The application defines the possible representations for each object. When a tool places an object in the object manager, it must also define various representations for it. The notebook then uses a reference to the object to request the representation from the object manager and includes it in the current document. Figure 3 shows an instance of this notebook.

4.2 Templates

The notebook, as well as the various agents communicate with each other using templates. We use the term *template* here in a somewhat broader sense than it has been used in the literature dealing with the user interfaces. Our templates include not only data requested/supplied by the user but also actions and abilities to deduce parts of the missing data in order to complete the user's requests. The templates aid the *proxy* to match words and phrases from some restricted natural language into applications, their parameters, locations (files), requests, etc. For example, if the user writes "solve PDE", the notebook

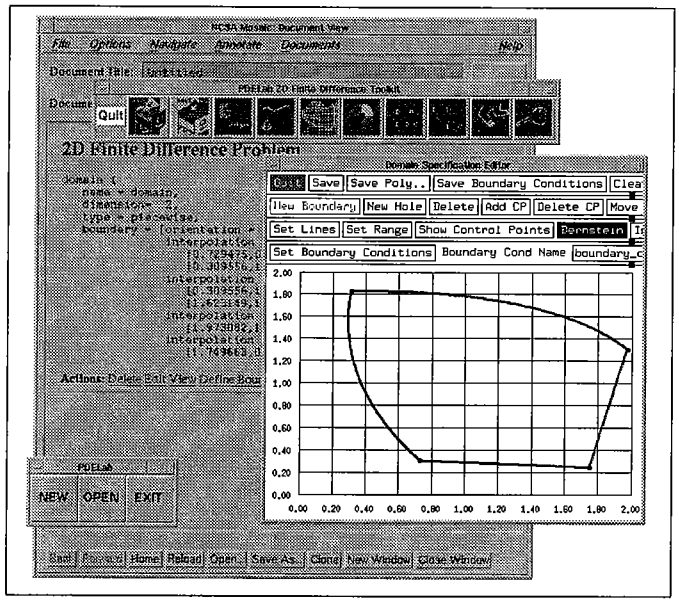

Figure 3: An instance of the *SciencePad* notebook.

manager invokes the PDE–solver *proxy*. This fills in the corresponding templates from the rest of the user's notes . If the user wishes to use, say, as a right–hand side of the equation the values from some experiment, he will identify the experiment (the lab, equipment, time, etc.) and the proxy will figure out where (in which file) the results are stored, approximate them if necessary, and incorporate the function into the equation to be solved. We call such an approach "template–driven" in the sense that the system tries to extract the maximum possible information from the input by identifying and filling in the appropriate template. From the users point of view, interaction with the system is like a sequence of high level instructions (natural language pseudocode, for instance). Although the templates will contain a lot of information, large parts of it could be considered default values. Then by using the templates at both (mobile and static) ends of the applications, we can significantly reduce the traffic over the wireless link by transmitting only those entries which are supplied by the user. In other words, templates also provide a well–defined interface between the

front–end (proxy) and the back–end of an application's interface.

4.3 PYTHIA– An Intelligent Agent to Guide PDE Solving

PYTHIA attempts to solve the problem of determining an optimal strategy (i.e., a solution method and its parameters) for solving a given PDE problem within user specified resource (i.e., limits on execution time and memory usage) and accuracy requirements (i.e., level of accuracy). PYTHIA uses the performance behavior of solution methods on previously solved problems as a basis for predicting a solution strategy for solving a given problem. The basic premise in PYTHIA's reasoning strategy is that performance data gathered during the solution of a set of problems using different solution methods can be used to predict the performance of these methods on a new problem. Of course, for this strategy to be correct, the new problem must be "similar" (i.e., have similar characteristics) to the problems that have been solved before. However, many of the steps involved in the process described above do not lend themselves to the kind of crisp decision making that a conventional strong AI approaches provide. More recent approaches, covered under the broad ambit of computational intelligence, may be used to give better results in such situations. With this in view, attempts are now on to incorporate techniques like neural networks into the PYTHIA framework. Connectionist techniques also learn from exemplars, but exhibit properties like generalization which are of singular interest in this situation. In other words, they seem to internally extract characteristics of the examples that they are seeing to group them into classes. This enables them to perform very well in classifying novel inputs. They also lend themselves to easy parallelization on SIMD architectures. This means that once their training phase is over, their classification is virtually instantaneous.

Some preliminary work done in this connection has produced very encouraging results. Specifically, backpropagation based networks have been used to identify the class to which a novel problem belongs [5]. Results from this experiment have been very encouraging, with both the learning and generalization abilities of the network clearly coming through. PYTHIA is now being extended to decide machine choice and configuration, as well as the parameters of the numerical methods using connectionist & fuzzy techniques.

5 Conclusion

In this paper, the authors describe *SciencePad*, a mobile platform to provide scientific computing power. The need for an intelligent information management system to achieve ubiquity is discussed, and the scenario for mobile computing is presented. We describe the software architecture of *SciencePad* and show how this architecture, with the use of AI techniques, addresses the problems that arise in designing a system that is mobile and accessible to the layperson. We also provide details of a prototype implementation of *SciencePad.*

References

[1] J.A. Berninger, R.D. Whitley, X. Zhang, and N.-H.L. Wang, *A Versatile model for simulation of reaction and nonequlibrium dynamics in multicomponent fixed–bed adsorption processes*, Computers in Chemical Engineering **15** (1991), no. 11, 749–768.

[2] Ahmed K. Elmagarmid, Jiansan Chen, Du Weimin, Omran Bukhres, and Rob Pez-

zoli, *Interbase: an execution environment for global applications over distributed, autonomous and heterogeneous software systems*, Tech. Report CSD-TR-92-016, Department of Computer Sciences, Purdue University, 1992.

[3] R. Fritzson, T. Finin, D. McKay, and R. McEntire, *Kqml- a language and protocol for knowledge and information exchange*, Proc. 13th International Distributed Artificial Intelligence Workshop, July 1994.

[4] E. N. Houstis and J. R. Rice, *Parallel ELLPACK: A development and problem solving environment for high performance computing machines*, Programming Environments for High–Level Scientific Problem Solving (P. W. Gaffney and E. N. Houstis, eds.), North–Holland, 1992, pp. 229–243.

[5] Anupam Joshi, Sanjiva Weerawarana, and Elias N. Houstis, *Using neural networks to support intelligent scientific computing*, International Conference on Neural Networks, IEEE, 1994.

[6] Sanjiva Weerawarana, *Problem solving environments for partial differential equation based applications*, Ph.D. thesis, Department of Computer Sciences, Purdue University, August 1994.

[7] Mark Weiser, *The computer for the twenty-first century*, Sci. Am. (1991), 94–104.

[8] ———, *Some computer science issues in ubiquitous computing*, Communications of the ACM **36** (1993), no. 7, 75–85.

[9] ———, *The world is not a desktop*, ACM Interations (1994), 7–8, personal communication.