# COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION

PROGRAM ANNOUNCEMENT/SOLICITATION NO./CLOSING DATE/if not in response to a program announcement/solicitation enter NSF 00-2

**NSF 98-168** **07/01/00**

FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) (Indicate the most specific unit known, i.e. program, division, etc.)

**ACI - ADVANCED COMP RESEARCH PROGRAM**

**FOR NSF USE ONLY**

**NSF PROPOSAL NUMBER**

**0086962**

| DATE RECEIVED | NUMBER OF COPIES | DIVISION ASSIGNED | FUND CODE | DUNS# (Data Universal Numbering System) | FILE LOCATION |
|---|---|---|---|---|---|
| | | | | 070710447 | |

EMPLOYER IDENTIFICATION NUMBER (EIN) OR TAXPAYER IDENTIFICATION NUMBER (TIN)

**351052049**

SHOW PREVIOUS AWARD NO. IF THIS IS
☐ A RENEWAL
☐ AN ACCOMPLISHMENT-BASED RENEWAL

IS THIS PROPOSAL BEING SUBMITTED TO ANOTHER FEDERAL AGENCY? YES ☐ NO ☒ IF YES, LIST ACRONYMS(S)

NAME OF ORGANIZATION TO WHICH AWARD SHOULD BE MADE

**Purdue Research Foundation**

AWARDEE ORGANIZATION CODE (IF KNOWN)

**0018259001**

ADDRESS OF AWARDEE ORGANIZATION, INCLUDING 9 DIGIT ZIP CODE

**Purdue Research Foundation**
**Hovde Hall**
**West Lafayette, IN. 47907**

NAME OF PERFORMING ORGANIZATION, IF DIFFERENT FROM ABOVE

**Purdue University**

PERFORMING ORGANIZATION CODE (IF KNOWN)

**0018259000**

ADDRESS OF PERFORMING ORGANIZATION, IF DIFFERENT, INCLUDING 9 DIGIT ZIP CODE

**Purdue University**
**West Lafayette, IN 47907**

IS AWARDEE ORGANIZATION (Check All That Apply)
(See GPG II.D.1 For Definitions) ☐ FOR-PROFIT ORGANIZATION ☐ SMALL BUSINESS ☐ MINORITY BUSINESS ☐ WOMAN-OWNED BUSINESS

TITLE OF PROPOSED PROJECT **Planning and Workflow Management for A Virtual Laboratory for Structural Biology**

| REQUESTED AMOUNT | PROPOSED DURATION (1-60 MONTHS) | REQUESTED STARTING DATE | SHOW RELATED PREPROPOSAL NO., IF APPLICABLE |
|---|---|---|---|
| $ **1,026,113** | **48** months | **01/01/01** | |

CHECK APPROPRIATE BOX(ES) IF THIS PROPOSAL INCLUDES ANY OF THE ITEMS LISTED BELOW

☐ BEGINNING INVESTIGATOR (GPG 1.A.3)
☐ DISCLOSURE OF LOBBYING ACTIVITIES (GPG II.D.1)
☐ PROPRIETARY & PRIVILEGED INFORMATION (GPG II.D.10)
☐ NATIONAL ENVIRONMENTAL POLICY ACT (GPG II.D.10)
☐ HISTORIC PLACES (GPG II.D.10)
☐ SMALL GRANT FOR EXPLOR. RESEARCH (SGER) (GPG II.D.12)
☐ VERTEBRATE ANIMALS (GPG II.D.12) IACUC App. Date _____
☐ HUMAN SUBJECTS (GPG II.D.12)
Exemption Subsection _____ or IRB App. Date _____
☐ INTERNATIONAL COOPERATIVE ACTIVITIES: COUNTRY/COUNTRIES _____
☐ FACILITATION FOR SCIENTISTS/ENGINEERS WITH DISABILITIES (GPG V.G.)
☐ RESEARCH OPPORTUNITY AWARD (GPG V.H)

PI/PD DEPARTMENT
**Department of Computer Science**

PI/PD FAX NUMBER
**765-494-0739**

PI/PD POSTAL ADDRESS
**1398 Computer Science Building**

**West Lafayette, IN 479071398**
**United States**

| NAMES (TYPED) | High Degree | Yr of Degree | Telephone Number | Electronic Mail Address |
|---|---|---|---|---|
| PI/PD NAME **Dan C Marinescu** | **PH.D** | **1975** | **765-494-6018** | **dcm@cs.purdue.edu** |
| CO-PI/PD | | | | |
| CO-PI/PD | | | | |
| CO-PI/PD | | | | |
| CO-PI/PD | | | | |



Received :05/16/2000

# A. PROJECT SUMMARY

We propose to create an infrastructure, a Virtual Laboratory, to support sharing of knowledge in structural biology, to plan an experiment, and to facilitate a rigorous process of scientific discovery. We call it a virtual laboratory because it will allow an individual sitting in front of a computer connected to the Internet, to control an experiment, to follow the most relevant aspects of an experiment conducted in the past by a well established group, or to get advice at a critical junction.

The process of discovery in structural biology, and other natural sciences like physics or chemistry is fairly complex, it involves some tedious and time consuming activities, sharing of knowledge, analysis of the results, backtracking. The virtual laboratory will allow structural biologists to reduce the time to obtain the results, improve the quality of the results, and restructure the human involvement by allowing a scientist to concentrate on the experiment and the discovery process. To reduce the time to obtain results we need efficient components, fully integrated experiment, simulation, and modeling environments, and some form of knowledge sharing and management. To improve the quality of the solution we need to fully integrate sensors in the feedback loop and to use optimal model parameters. Last but not least, we need to automate simple tasks like data migration, format conversion, and selection of optimal running condition for each computational task. The high level of sophistication necessary to obtain quality results in this field motivates our effort.

Some components of the infrastructure are specific to structural biology e.g. the interfaces for building the knowledge base, but the basic mechanisms e.g., the planning and the workflow enactment engines, are general and can be used for other classes of problems with similar characteristics. In this grant application we propose to develop agents for 3D structure determination using cryo TEM methods and to design an integrated workflow enactment engine based upon a Petri Net model with a planning engine and a knowledge management system.

All our programs are distributed under open source licenses from our Web site: http://bond.cs.purdue.edu. The project will contribute to education and support knowledge sharing in structural biology laboratories.

# TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.


| Section | | Total No. of Pages in Section | Page No.* (Optional)* |
|---|---|---|---|
| Cover Sheet (NSF Form 1207) (Submit Page 2 with original proposal only) | | | |
| A | Project Summary (not to exceed 1 page) | 1 | |
| B | Table of Contents (NSF Form 1359) | 1 | |
| C | Project Description (plus Results from Prior NSF Support) (not to exceed 15 pages) **(Exceed only if allowed by a specific program announcement/solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)** | 15 | |
| D | References Cited | 7 | |
| E | Biographical Sketches (Not to exceed 2 pages each) | 0 | |
| F | Budget (NSF Form 1030, plus up to 3 pages of budget justification) | 6 | |
| G | Current and Pending Support (NSF Form 1239) | 1 | |
| H | Facilities, Equipment and Other Resources (NSF Form 1363) | 1 | |
| I | Special Information/Supplementary Documentation | 0 | |
| J | Appendix (List below. ) **(Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)** | | |
| | Appendix Items: | | |


*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated. Complete both columns only if the proposal is numbered consecutively.

# C. PROJECT DESCRIPTION

## C.1 RESULTS FROM PRIOR NSF SUPPORT

Dan C. Marinescu is the PI of the NSF Grand Challenge Award MCB-9527131, Parallel and Distributed Computing for Solving Large Structural Biology Problems. He was the co-PI of the research grants CCR-9119388 and BIR-9301210. These grants provided support for four post-doctoral fellows, Drs. Zhongyun Zhang, K.C. vanZandt, Hong Lin, and Ruibin Hao. The Ph.D. dissertations of Marius Cornea-Hasegan, Ioana Martin, Kuei Yu Wang, Mihai Sirbu, and Ladislau Bölöni, were supported by these grants. Three Ph.D. students, Kyungkoo Jun, Krzysztof Palacz and Radu Sion are currently supported. We developed parallel algorithms and programs for the study of the 3D structure of biological macromolecules and viral assemblies using structural information obtained through X-ray diffraction. The parallel programs have been used since 1992 by Michael G. Rossmann, John E. Johnson and others for the determination of the structure of several viruses. Two graphics packages TONITZA and EMMA, are used by structural biologists at the Scripps Institute in San Diego, NIH, Karolinska Institute in Sweden, and a research institute in Helsinki, Finland. We have also distributed parallel programs for 3D reconstruction of viruses and for orientation determination for electron microscopy. These programs are available for downloading from http://www.cs.purdue.edu/homes/sb. The Bond agent system http://bond.cs.purdue.edu was released under an open source license, in March 1999 and has been downloaded by more than 500 sites.

### Publications (chronological)


[1] Marinescu, D. C., J. R. Rice, M. A. Cornea-Hasegan, R. R. Lynch, and M. G. Rossmann (1993). Macromolecular electron density averaging on distributed memory MIMD systems. Concur.: .P&E, 5:635–657.

[2] Cornea-Hasegan, M. A.(1994). Determination of biological macromolecular structures using distributed memory MIMD systems. Ph.D. Thesis, Purdue University.

[3] Cornea-Hasegan, M. A., D. C. Marinescu, and Z. Zhang (1994). Data management for a class of iterative computations on distributed memory MIMD systems. Concur. Prac. and Exper. 6:205–225..

[4] Marinescu, D. C. and J. R. Rice (1994). On high level characterization of parallelism. J. of Paral. and Distrib. Computing 20:107–113.

[5] Marinescu, D. C. and J. R. Rice (1994). On the scalability of asynchronous parallel computations. J. of Paral. and Distrib. Computing 22:538–546.

[6] Cornea-Hasegan, M. A., Z. Zhang, R. E. Lynch, D. C. Marinescu, A. Hadfield, J. K. Muckelbauer, S. Munshi, L. Tong, and M. G. Rossmann (1995). Phase refinement and extension by means of non-crystallographic symmetry averaging using parallel computers. Acta Cryst. D51:749–759.

[7] Martin, I. M., D. C. Marinescu, and J. R. Rice (1995). Adaptive load balancing strategies for solving irregular problems on distributed memory MIMD systems. Proc. IPPS 95, IEEE Press, pp. 57–64.

[8] Martin I.M. and D.C. Marinescu (1996). Exploiting symmetry in parallel computations for structural biology. Proc. Euro-Par'96, LNCS, Vol. 1124, Springer Verlag, 255-259.

[9] Martin, I. M. (1996). Scientific data visualization and digital image processing in structural biology. Ph.D. Thesis, Purdue University.

[10] Sirbu, M. and D.C. Marinescu (1996). Bond – a parallel virtual environment. Proc. HPCN 96, High Performance Computing and Networking, LNCS, Vol. 1067, Springer Verlag, 255-259.

[11] Wang, K. Y. and D.C. Marinescu (1996). Empirical studies of paging and I/O activity of parallel programs. Proc. MASCOTS'97, IEEE Press, 177-184.

[12] Wang, K. Y (1996). Hiding the latency of paging and I/O operations on massively parallel systems. Ph.D. Thesis, Purdue University.

[13] Costian, C. and D. C. Marinescu (1996). A distributed memory algorithm for 3D FFT. Journal of Computational and Applied Mathematics, 66, 139-151.

[14] Martin, I. M., D. C. Marinescu, R. E. Lynch., and T. S. Baker (1997). Identification of spherical virus particles in digitized images of electron micrographs. J. Struct. Biol. 120, 146-157.

[15] Sirbu, M.G. (1997) The design of a meta-computing environment. Ph.D. Thesis, Purdue University.

[16] Martin, I. M. and D. C. Marinescu (1998). Combining visualization and computations for interactive data analysis of biology data. Proc. IASTED Conf. on Comp. Graphics, IEEE Press, 179-182.

[17] Martin, I.M. and D.C. Marinescu (1998). Concurrent computations and data visualization for spherical virus determination. IEEE Computational Science & Engineering, October-December, 40-52.

[18] Wang, K. Y., D.C. Marinescu, and O.F. Carbunar (1998). Dynamic scheduling of process groups. Concurrency: P& E 10(4): 265–283.

[20] Bölöni L. and D.C. Marinescu (1999). On the robustness of metaprogram schedules.

Proc Heterogeneous Computing Workshop, HCW'99, IEEE Press, 146–155.

[21] Bölöni L. and D.C. Marinescu (1999). Three theorems on robustness of metaprogram schedules. Proc. ACM ICS Workshop on Scheduling Algorithms, Rhodes, IEEE Press.

[22] Bölöni L., R. Hao, K.K. Jun, and D.C. Marinescu (1999). Structural biology metaphors applied to the design of a distributed object system. Proc. Second Workshop on Bio-Inspired Solutions to Parallel Processing Problems, LNCS, Vol. 1586, Springer Verlag, 275–283.

[23] Lynch R.E., D.C. Marinescu, H. Lin, and T.S.Baker (1999). Parallel algorithms for reconstruction of asymmetric objects from electron micrographs. Proc. IPPS 99, IEEE Press, 632-637.

[24] Bölöni L (2000). Contributions to distributed objects and network agents. Ph.D. Thesis, Purdue University.

[25] Marinescu, D.C, and L. Bölöni (2000). Biological metaphors in the design of complex software systems. Journal of Future Generation Computing Sustems, Elsevier, (in press).

[26] Marinescu, D.C. (2000). An agent-based design for Problem Solving Environments. Proc. Workshop on Parallel/High Performance Scientific Computing, LNCS, Springer Verlag, (in press).

[27] Bölöni L., D.C. Marinescu, J.R. Rice, P. Tsompanopoulou, and M. Vavalis (2000). Agent based networks for scientific simulation and modeling. Concurrency: P&E, Willey, 2000 (in press).

[28] Bölöni L. and D.C. Marinescu (2000). An object-oriented framework for building collaborative network agents. In Intelligent Systems and Interfaces (N.H. Teodorescu , D. Mlynek, A. Kandel, H.J. Zimmerman, eds.). Kluewer Publishing House, 31–65.

[29] Bölöni L. and D.C. Marinescu (2000). Robust scheduling of metaprograms. Journal of Scheduling, Willey, 2000 (in press).

[30] Bölöni L. and D.C. Marinescu (2000). A component-based agent model: from theory to implementation. Proc. Second Int. Workshop "From Theory to Agent Implementation", 2000 (in press).

[31] Bölöni L. and D.C. Marinescu (2000). A multi-plane agent model. Proc. Autonomous Agents 2000, Barcelona, June 2000 (in press).

[32] Bölöni L., K.K. Jun, K. Palacz, R. Sion and D.C. Marinescu (2000). The Bond agent system and Applications. Proc. ASA/AM 2000, LNCS, Springer Verlag, 2000 (in press)

[33] Bölöni L. and D.C. Marinescu (2000). Agent surgery: the case for mutable agents. Proc. Third Workshop on Bio-Inspired Solutions to Parallel Processing Problems, LNCS, (in press).

[34] Palacz K and D.C. Marinescu (2000). An agent-based workflow management system. Proc. Workshop Bringing Knowledge to the Business Process, Stanford University, AAAI Press, 119–127.

[35] Jun K.K., L. Bölöni, D.K.Y. Yau, and D.C. Marinescu (2000). Intelligent QoS support for an adaptive video service. Proc. IRMA 2000, International Resource Management Association (in press).

[36] Jun K.K., L. Bölöni, K. Palacz, and D.C. Marinescu (2000). Agent-based resource discovery. Proc Heterogeneous Computing Workshop, HCW'00, IEEE Press (in press).

[37] Lynch R.E., H. Lin, and D. C. Marinescu (2000). An efficient algorithm for parallel 3D reconstruction of asymmetric objects, from electron micrographs, *Proc. Euro-Par 2000*, LNCS, Springer Verlag 2000 (to appear).

## C.2 DESCRIPTION OF THE PROPOSED RESEARCH

### C.2.I A Virtual Laboratory for Structural Biology.

We propose to create an infrastructure, a Virtual Laboratory, to support sharing of knowledge in structural biology, to plan an experiment, and to facilitate a rigorous process of scientific discovery. We call it a virtual laboratory because it will allow an individual sitting in front of a computer connected to the Internet, to control an experiment, to follow the most relevant aspects of an experiment conducted in the past by a well established group, or to get advice at a critical junction. The infrastructure will support mundane tasks like computing on a cluster of workstations, data staging and so on. Though at the time of this writing we do not have any plans to support remote experiments per se, to allow individuals to submit their samples and then to control the microscope, the framework we are developing is entirely capable to do so. The virtual laboratory will allow us to reduce the time to obtain the solutions, improve the quality of the results, and restructure the human involvement by allowing a scientist to concentrate on the experiment and the discovery process, Figure 1b. A number of problem solving agents will be designed.

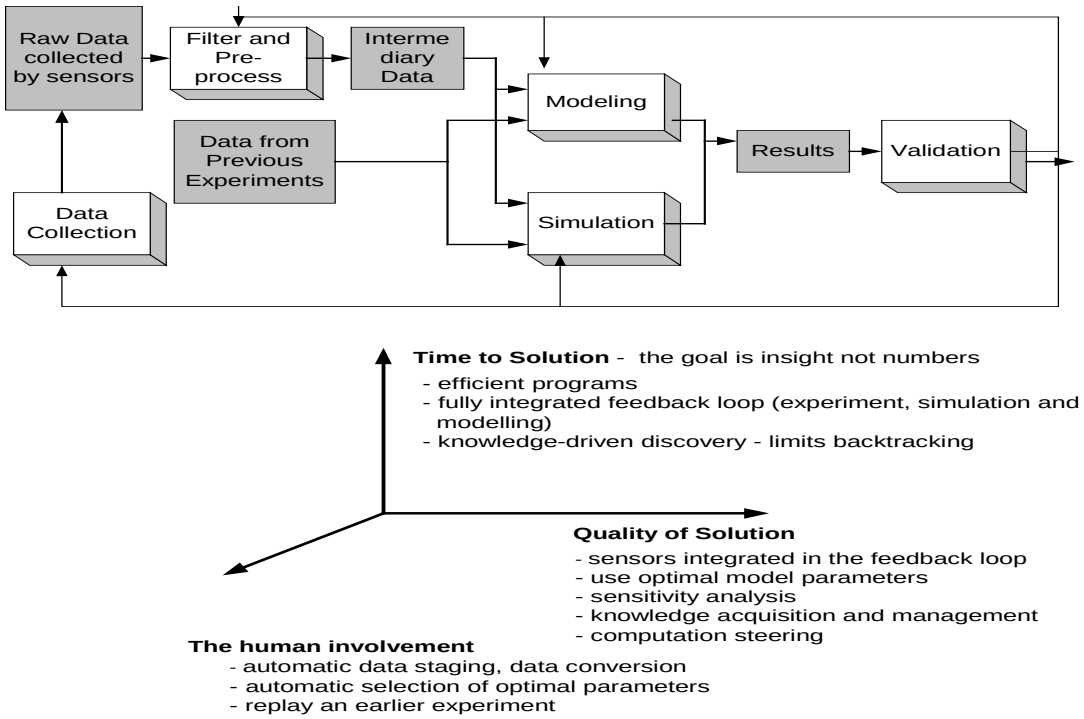



**Figure 1**. (a) The data flow in computational biology, common to many data intensive applications. (b) The three goals of a Virtual Laboratory: (b.1) reduce the time to solution, (b.2) improve the quality of the solution, and (b.3) refocus the human involvement.

Some components of the infrastructure are specific to structural biology e.g. the interfaces for building the knowledge base, but the basic mechanisms e.g., the planning engine, are general and can be used for other classes of problems with similar characteristics. In our model presented in Figure 1a, a group of individuals collaborate to solve a problem. They carry out experiments and collect data produced by sensors connected to the experimental setup. The amount of experimental data is very large; the user needs to keep a detailed log of the experimental conditions and the precise setup used for each set of data. The next step is to extract the relevant information from the experimental data. Seldom this process is straightforward; often it requires a significant computational effort. Once the experimental data are distilled, they are plugged into a computational model. This model itself is typically very complex, depends upon a large number of parameters and computations have to be carried out repeatedly with different sets of parameters until the results are deemed to be acceptable. Sometimes the entire process takes months if not years and involves complex interactions between humans and between humans and

computers. The members of a group are often geographically distributed and have to use a diversity of computers.

Software agents may provide an answer to the increased complexity of the software systems expected to intelligently anticipate and adapt to the needs of dynamically distributed applications as indicated in Figure 1b. The primary functions of agents in the Virtual Laboratory we propose are: knowledge management, planning, scheduling and control, resource discovery, management of local resources, use-level resource management, see Figure 2. Several components of the project are already in place, we outline the significant features of the Bond system and then discuss the knowledge management and experiment planning components. A microserver [17] allows control of agents from the Web.

***Motivation – the high level of sophistication necessary to obtain quality results.*** The process of discovery in structural biology, like in other natural sciences like physics or chemistry is fairly complex, it involves some tedious and time consuming activities, sharing of knowledge, analysis of the results, backtracking. Structural biology is an experimental science. In addition to structural biology knowledge it requires understanding of the physical phenomena used to gather experimental data, e.g. electron and X-ray diffraction or NMR, mastering of techniques to prepare samples and to conduct an experiment, familiarity with computational methods to extract information from experimental data and plug it in into computational models, and, the ability to interpret the results obtained at each step of the process. Several individuals with expertise in different fields need to work together to ensure the success of an experiment. New techniques to reduce the time required for a certain step, or automate tedious activities have to be assimilated continuously. There is a price to pay for these advancements, the level of sophistication required to obtain high quality results increases. The unsuspecting scientist may be unable to use these advancements without proper guidance, may get incorrect results due to incomplete understanding of the limitations of the new methods/techniques, or may simply be overwhelmed by the vast amount of data. Several illustrative examples follow.

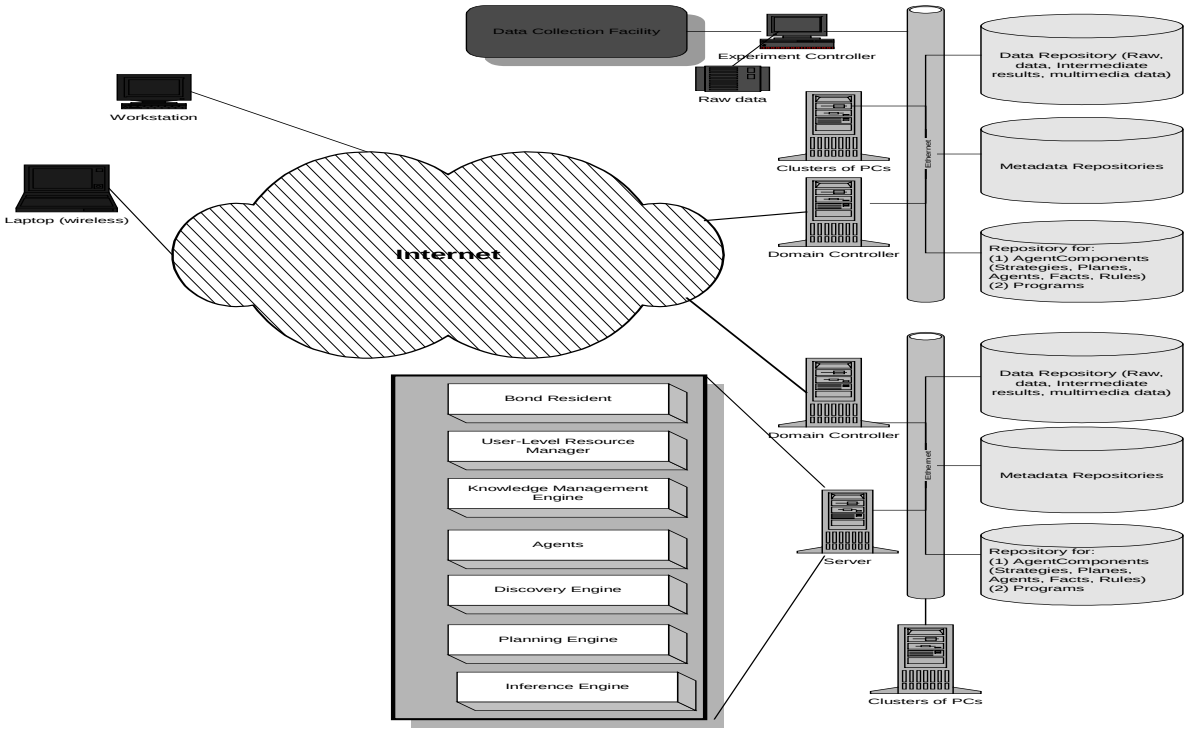



**Figure 2**. The Virtual Laboratory will allow Web-based access to computational resources.

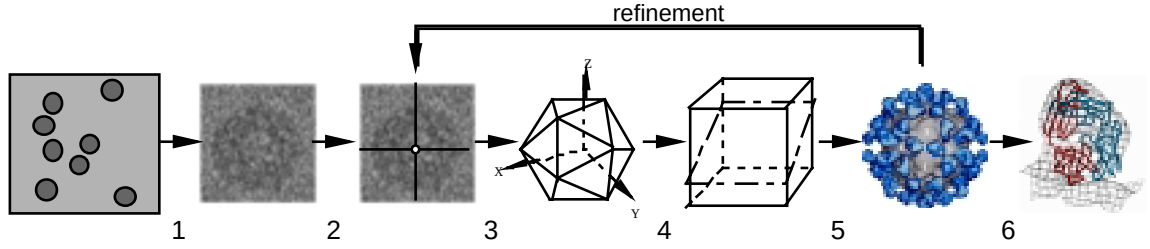
The use of CCDs instead of photographic emulsions to record diffraction images in cryo-TEM and crystallography illustrates the case of a new technique that opens new possibilities and generates new problems. CCDs create digital images that are immediately available for processing, a great advantage. On the other side of the coin, the high data acquisition rates imply that data must be properly organized and cataloged and metadata describing the setup of the microscope, the temperature, the sample used for each image, must be created. With thousands micrographs collected in a few days it is no longer feasible to maintain a hardcopy log and we face the need for an archival system for raw data and intermediate results. CCDs may lead to another qualitative improvement. If we can control the microscope in real-time and automate the particle location process we will be able to generate higher contrast images using exposures with different levels of radiation. In the first phase we expose the sample to a low intensity beam, then we locate a particle, and finally we focus a high intensity beam on the region of the sample we suspect the particle to be located and obtain a high contrast image.

## C.2.II. LEVERAGE.

We are in a unique position due to our long-term collaboration with renowned structural biologists and to our work on software agents. During the past twelve years we have designed individual components as well as the kernel of the virtual laboratory for structural biology. We designed parallel algorithms and programs for processing X-ray diffraction data and for 3D reconstruction for cryo-electron microscopy as well as the Bond agent infrastructure.

### *A. Algorithms and programs for cryoTransmission Electron Microscopy.*

The advantages of using cryoTEM and 3D reconstruction techniques to explore details of viral pathogenesis have clearly prompted keen interest from scientists worldwide. The number of structural studies of icosahedral viruses by cryoTEM, and 3D image reconstruction has mushroomed over the past decade. However, the number of research laboratories actively pursuing these studies has remained quite low owing to the critical need for sophisticated equipment and highly trained personnel. Figure 3 illustrates the processing pipeline in a cryoTEM experiment.




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Current | n= 100 | 1-2 hrs | mins | hrs | mins | mins | hrs/weeks |
| | n=2000 | days | mins/hrs | days | mins/hrs | hrs/day | hrs/weeks |
| Goal | n= 100 | secs | ~1 sec | secs | < 10 secs | < 10 secs | secs/hrs |
| | n=2000 | mins | < 1 min | mins | < 1 min | < 10 mins | mins/hrs |

**Figure 3**: Schematic representation of the processing 'pipeline' from 2-D images to 3-D modeling. Step 1: extract individual particle images from electron micrograph or CCD image. Step 2: determine initial location of each particle center. Step 3: determine each particle view orientation. Step 4: fill in 3-D transform. Step 5: compute 3-D reconstruction (shown is rhinovirus-Fab complex, with Fabs colored blue). Step 6: dock atomic model into cryoTEM 3-D density map (example shows an atomic antibody Fab model docked into the corresponding Fab portion of the virus-Fab 3-D reconstruction). Refinement involves using an intermediate 3-D reconstruction as a model to better define center (x,y) and orientation ($\theta,\phi,\omega$) parameters for all particle images, followed by Steps 4 and 5 and additional cycles as needed to include more particle images and extend resolution. Not depicted here are the steps involving specimen preparation, low-dose cryoTEM, and digitization of the images (all of which precede Step 1), which can take as few as several hours to complete for ideal samples, or, more typically, days or weeks for difficult specimens. Digitization may be performed at the microscope (secs-mins) by recording images on a slow-scan CCD camera or by scanning images recorded on photographic film with a microdensitometer (mins-hrs). Time estimates for use of current technology are compared to estimates based on achieving the goals of this proposal ($10^2$ to $10^3$ fold faster) for each step. The time frame increases approximately linearly with the number of cycles of refinement (Steps 2-5). Typically, four or less cycles are sufficient for processing 'good' data at 20Å resolution.

## A1. Image Processing: Automatic Particle Identification

The first steps in the 3-D reconstruction of a virus structure are designed to extract the motif representing the projection of a particle in micrographs of frozen-hydrated biological specimens and to obtain an initial estimate of the position of the center of each particle (Figure 3, Steps 1 and 2). At high magnification and low electron doses, noise in electron micrographs is unavoidable. Variability in the background of the support film and electron radiation damage to the specimen are the two major sources of noise. The biological specimen is exposed to low doses of radiation to minimize the radiation damage, but this leads to very low contrast images. Still, the human visual system is unsurpassed in its ability to analyze micrographs effectively and reliably, and to identify quickly virus particle projections even in noisy, low contrast images. However, even for low resolution image reconstructions that utilize only a hundred or fewer projections, this task is tedious, time consuming, and prone to errors (van Heel, 1982). Furthermore, even trained users have a difficult time selecting accurately the positions of the particle centers in noisy images (Fig.3, Step 2). For high resolution work requiring thousands or more particle projections, the task of particle extraction from micrographs becomes prohibitive and prone to error when performed manually. One might wonder if such a task can be totally automated. That is unlikely because it is unreasonable to believe that one can design image-processing algorithms that are able to identify virus particles with a high success rate, regardless of the quality of the micrograph. Yet we can train a recognition algorithm to work well with images of a certain quality by setting up several parameters of the algorithm [90, 91]. Once we have established these parameters, successive micrographs taken under similar conditions can be processed with minimal if any human intervention.

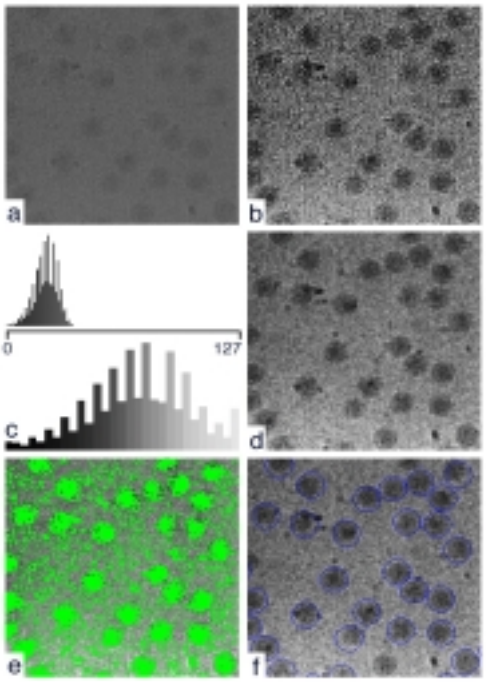



**Figure 4**: The basic features of the CP2 algorithm for automatic particle identification we introduced in [91]. (a) Portion of low contrast micrograph of frozen-hydrated sample of reovirus cores. (b) The micrograph after histogram equalization. (c) Gray level histograms before (top) and after (bottom) histogram equalization. (d) The micrograph in (b) after neighborhood averaging with a 10 x 10 filter. (e) Contents of the binary image after pixel marking (green) superimposed on the micrograph in (d). (f) The result of the CP2 method, [91].

***A.2 Parallel algorithms for 3D reconstruction and orientation determination.*** The 3D reconstruction is an iterative process that starts with an initial electron density map and goes through a number of iterations consisting of orientation determination followed by 3D reconstruction. We have developed parallel algorithms for orientation determination and 3D reconstruction and have been able to speed up the process considerably [77-79]. For example one iteration of the 3D reconstruction for the Bursalia Corella Virus that used to take about 4 hours using a sequential program was carried out in less than 3 minutes on 16 nodes of a PC cluster, using the program based upon our algorithm.

***B. Bond Agent framework***. For the past three years we developed a distributed-object, message-oriented system and a constructive framework for collaborative network agents [8], [13], [15-18]. At the time of this writing Bond consists of about 90,000 lines of Java code, is released under an open source license, LGPL, and has already been downloaded by more than 500 individuals, from our site: http://bond.cs.purdue.edu. So far, the system has been used as a workflow enactment engine supporting dynamic workflows [102], for an adaptive video service [65], for resource discovery in a wide area distributed system [66], for the design of a network of PDE solvers [12], for teaching distributed system courses, and for several other applications.

## C.2.III. RELATED WORK.

A number of new initiatives and ideas for high performance distributed computing have emerged in the last years [6], [7], [25], [27], [33], [36], [37], [39], [44-47], [58], [91], [122], [132]. Object-oriented design and programming languages like Java open up intriguing new perspectives for the development of complex software systems capable to simulate physical systems of interest to computational sciences and engineering. The Java Grande initiative aims to add new constructs and to support optimization techniques needed to make the Java language more expressive and efficient for numerical simulation. If successful, this effort will lead to more robust scientific codes and increased programmer productivity. An important side effect of the use of Java in scientific computing is code mobility. This brings us to another significant development, *computing grids*. Informally, a computing grid is a collection of autonomous computing platforms with different architectures, interconnected by a high-speed communication network. Several research projects notably Legion [58] and Globus [44-46] are actively involved in designing the software support for computing grids. Many problems in computational sciences and engineering could benefit from the use of computing grids. Yet, scientific code mobility, a necessary condition for effective use of heterogeneous computing environments is a dream waiting to materialize. Porting a parallel program from one system to another, with a different architecture and then making it run efficiently are tedious tasks. Thus the interest of the high performance distributed computing community for Java. Our effort to integrate knowledge management in the system was inspired by the Waldo project at Berkeley [64]. Its goal is to build a real-time widely distributed instrumentation system using *data annotation* for data intensive applications.

There is a vast body of literature on software agents, [19-21], [23-24], [29], [42-42], [48-49], [52-54], [57], [69], [99-100], [103], [108,109], [117], [128,129] but precious little information about agents in high performance distributed computing.

## C.2.IV. SOFTWARE AGENTS FOR HIGH PERFORMANCE DISTRIBUTED COMPUTING.

A software agent is a program expected to exhibit to some degree, attributes broadly classified in three groups, see Bradshaw [20]:

- Agency: measures the degree of autonomy and authority of an agent. It reflects the nature of the interactions between an agent and: the user, other agents, data, and services.
- Intelligence: reflects the degree of preferences, reasoning, planning, and learning behavior.
- Mobility: the ability to travel through a network.

Agents able to meet the Turing test by emulating human behavior are useful for some applications in science and engineering, e.g., deep space explorations, robotics, and so on; they are investigated by the AI community. Our view of an agent is slightly different: an agent is an abstraction for building complex systems [11]. Our main concern is to develop a constructive framework for building collaborative agents out of ready-made components and to use this infrastructure for Problem Solving Environments, PSEs [87]. To use a biological metaphor, *software agents form a nervous system and perform command and control functions in a PSE* [86].

**Mixins of agents and legacy code.** The algorithms for data analysis are already very complex and adding resource management or adaptation logic would make them unnecessarily brittle and difficult to modify. Since Java is not yet suitable to write scientific software, we propose to create mixins consisting of legacy programs written in programming languages understood by structural biologists, FORTRAN and C, and agents made out of Java components, able to support self-scheduling and adaptation. The advantages of this approach are:

- Separation of concerns. The processing algorithms are created by a domain scientist, the control functions and structures by a computer scientist.
- Legacy codes require minor or no adaptation at all. This leads to a substantial reduction of the development time and to increased reliability.
- The resulting ensemble is more adaptive, more functional, and easier to use.

We also recognize potential problems with this approach. First, there is an additional overhead for communication and control functions. Agents are rather slow, they are often written in Java and some of their functions, e.g., the inference, require a fair amount of iteration. But the control functions are exercised seldom and in most cases the benefits of the agents outweigh by far the slight performance penalty. Second, though the agents are mobile, the legacy code is not. Porting a large legacy application to a new hardware architecture and operating system is a major endeavor.

Typically, an agent has as input a set of rules and facts. Some of the facts are rather static, e.g. those describing the configuration of the system or the characteristics of the problem, other are more dynamic e.g. partial results needed to make a decision for the next step, system load needed to adapt to the environment. The agent and the legacy application interact through a *problem description file* produced by the agent. This file contains the location of the data, parameters of the model, parameters of the algorithms, and hints. The function of the agent is thus to generate the problem description file using inference and then start up the legacy application and monitor it throughout its execution. The agent acts as a *wrapper* of the legacy application.

We have tested this approach and implemented a network of PDE solvers based upon legacy solvers developed at Purdue by John Rice's group. The results reported in [12] and [123] are extremely encouraging, a novice implemented the network of PDE solvers as a class project, and was able to develop coordination, mediator, and solver agents in less than one month, while the original effort was a multi year project leading to a Ph.D. dissertation.

**Bond Agents.** The agent infrastructure we have developed is based upon a *multi-plane* agent model described in detail in several publications, [13], [15], [16], [17]. Each plane reflects a facet of the agent activity, e.g. interaction with the user, control of an application, etc. In each plane the agent behavior is described by a state machine; once an agent enters a state it executes a *strategy*, typically a Java program. Strategies communicate with one another using a shared memory called *the model of the agent*. Planes and strategies are reusable components used to assemble an agent. Examples of strategies are data staging used to transfer a file from one platform to another, an inference strategy based upon the Jess expert

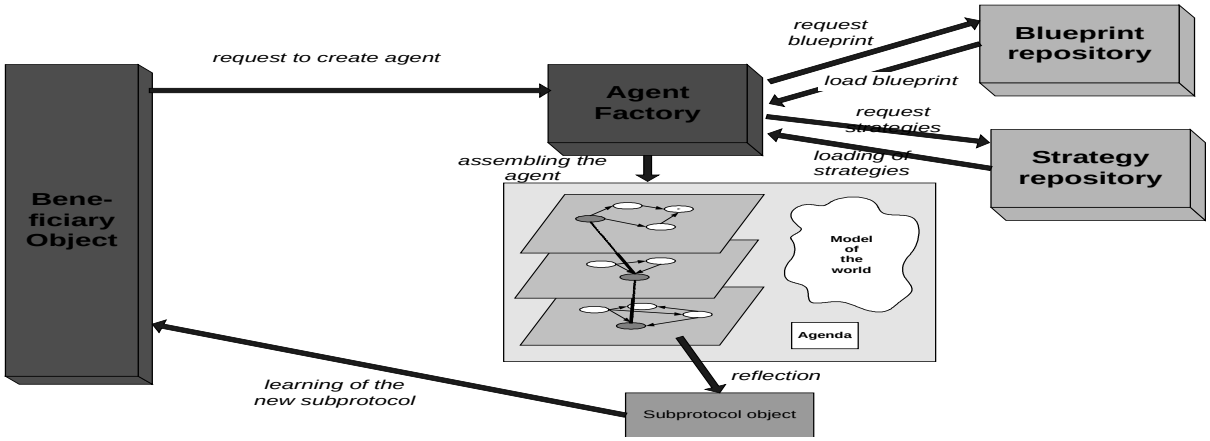



**Figure 5**. Agent creation in Bond. The beneficiary sends to the agent factory a control message in the agent control subprotocol. The blueprint or the URL of the blueprint in a repository, are provided in the message. The agent factory assembles the agent using the strategy repositories. The agent has multiple planes, each a finite state machine.

system shell, developed at Sandia National Laboratory [49], a remote execution strategy used to start up an executable on a remote system, and so on. At the time of this writing Bond strategy repositories have close to fifty strategies. Agents communicate with one another using messages. Bond supports messages in KQML, Knowledge Query and Manipulation Language, [41], [42] or XML. Closed sets of messages form a *subprotocol* and are used to carry out a specific function, e.g the agent control subprotocol consists

of messages to create, start, stop, and control agents. Several communication engines, one based upon TCP, one on UDP, one on the Infospheres, [27], and one based upon multicast can be used to transport messages from one Bond resident to another. A *resident* is a container, has a local directory and a communication engine and can load dynamically other objects, for example, the agent factory discussed below. The structure of an agent is described in an agent description language called *Blueprint*. An XML extension of the Blueprint is planned. *Agent factories* assemble agents from their textual description in Blueprint and control their execution, see Figure 5. An agent is represented at run time by a data structure reflecting the state and the internal structure of the agent.
The system supports *weak agent migration* [13], [15], and *agent surgery* [18]. To migrate an agent the beneficiary (another agent or the human) sends a control message to the agent factory to stop a running agent. Since multiple planes are running concurrently the agent will be stopped only after the running strategies, one in each plane, finish their execution. Then the beneficiary sends to the agent factory at the new site the blueprint together with the model, which contains the current state, the agent is reassembled and started. Surgery is the process of modifying at run time the data structure controlling the agent. Once an agent is modified, a blueprint reflecting the changes can be generated. Surgery is useful for trimming an agent before migration; states unreachable from the current state are deleted to reduce the footprint of the agent. Agent mobility is a useful feature, particularly for data intensive applications. Often, it is more efficient to send an agent to the data collection point to check the consistency of the data rather than transfer a large amount of data to the processing site through a network with limited bandwidth. Agent security is enforced by means of *security probes* as described in [60] and [61]. Scalability studies are provided in reference [24] in Section C.1 and indicate that the system is capable to handle tens of thousands of objects at each site and that the overhead for object creation and communication is comparable with that measured by several CORBA implementations [55].

**Agents for data analysis in cryoEM.** Countless examples of challenges encountered by a structural biologist can be given, but we choose only two examples to illustrate the increased computational sophistication needed for cryo-TEM, one in the area of image processing and the other one in the area of parallel computing. We plan to design an agent for each of the tasks described below.
**Example 1**. Identification of virus particles in a micrograph is an example of a tedious activity a human was able to carry out when a few hundred projection were sufficient for 3D reconstruction of a symmetric structure at low resolution, but becomes unfeasible when the number of projections increases by two orders of magnitude as needed for high resolution studies of asymmetric structures. To increase the resolution of the structure determination from the 20 to 5 Å the number of projections is estimated to increase from few hundreds to 20 to 50,000. Several groups including ours have proposed methods to automate the process. Incidentally, our method described in [90] produces considerably fewer false hits, around 2-3%, than others who typically result in 30-40 % false hits. Figure 4 illustrates the basic steps of our algorithm: (1) image enhancement including histogram equalization and averaging, (2) marking, a step to identify pixels inside a virus projection, (3) clustering of marked pixels and elimination of clusters that are too large or too small, (4) center identification by determining the center of the mass of a cluster and finally, (5) the refinement of particle centers by moving the center of the particle within a refinement box and correlating the resulting particle with a model particle obtained by averaging all particles. But the quality of the solution of the automatic particle recognition in our algorithm depends upon the ability of the end user to provide accurate information for the recognition algorithm, e.g., the radius of the particle, the number of thinning layers needed to separate clusters, the parameters of the histogram equalization algorithm, the size of the region used for averaging during the image enhancement phase, and so on. The function of the agent is to run the algorithm with one set of parameters and then to repeat the measurement with a different set and then choose the optimal values of the parameters. An agent can carry out this process because we can clearly formulate a goal function, namely minimization of the correlation coefficients between the model particle and individual projections. On the other hand, it is very difficult for a structural biologist to understand all the subtleties of the algorithm and the effect of its

internal parameters. As a result, the quality of the solution produced when a novice runs the program is inadequate and the entire 3D reconstruction process starts with noisy data.

**Example 2**. Now consider the parallel algorithm for orientation determination mentioned earlier. The idea of the algorithm, described in detail in [77], is to construct a data base of calculated projections using as input the current value of the electron density map and then to compare each image of a virus projection obtained experimentally, against each calculated image in the database, whose orientation is known. We assign the orientation of the experimental projection based upon the correlation coefficient between the two. Thus the algorithm is straightforward and embarrassingly parallel. An image may consist of 0.5 MB of data and the database of calculated images may consist of about 4,000 images or about 2 GB for a virus with icosahedral symmetry because we only need electron density for a wedge of the virus volume, the so called asymmetric unit. For an asymmetric virus the database could be two orders of magnitude larger because we need the electron density data for the entire volume of the virus. Sometimes, the symmetry of the virus is not known and we are interested to determine the axes of symmetry. In this case we have to construct the full database of calculated images and for each experimental image determine its correlation coefficient with every single image in the upper hemisphere, then rank the correlation coefficients and deduce the symmetry. We outline some of the options we have and the decisions to be made. Our first algorithm is based upon a multi-resolution approach. For an icosahedral virus we first construct a low-resolution database (at 3 degre resolution we have only 59 images), find out the approximate orientation of each image, then we construct the high-resolution database (at say 0.3 degree resolution we have 3984 images) and conduct the search of the optimal orientation on it. Reference [91] provides detailed explanations for the numbers above and due to lack of space we cannot include a figure to illustrate the effect of the symmetry. The high-resolution database is distributed across nodes and the experimental images are sorted based upon the approximate orientation determined in the first phase. Each node will process only a fraction of the total number of experimental images. The choices to be made are:

- How to get the input data in each node. We can read the electron density and the experimental image files in parallel in each node, or had one node read the data and broadcasts it to the other nodes as needed.
- How to distribute the data and computations amongst nodes for the second phase. We have the choice of dividing the database of calculated images evenly among nodes, each node will compute a different segment of the data base. In this case the number of experimental images to be processed by each node is likely to be different, there is no guarantee that experimental images have uniformly distributed orientations, and the load is imbalanced. The alternative is to distribute evenly the number of experimental images but risk that the segment of the database assigned to one node is much larger than the one for other nodes.
- How to deal with a system consisting of a cluster of heterogeneous workstations shared among a group of users. The load distribution is affected by the relative speed of individual nodes and by the current load due to other processes.

The decisions above should be based upon the actual machine configuration, the presence of a parallel file system, the speed of the processor(s) and the amount of main memory in each node, the speed of the interconnection network, the architecture of the system, e.g., distributed memory or a set of shared memory nodes interconnected together, the actual load of each node. An optimal decision regarding load balancing can be made only at run time when the results of the global search mode are known. To balance the load one could use a heuristics based upon the time to calculate an image and the time to correlate a calculated image with an experimental one.

The problem is further complicated when we do not know the symmetry. The multi-resolution algorithm described above will not work for large viruses because at low resolution their features are completely lost. Thus for a large particle whose symmetry is not known we need to construct from the beginning a very large database and conduct a linear search for the optimal correlation coefficient throughout the entire database. Because of the sheer amount of data, 100 GB or more, we need a large system configuration and distribute calculated images evenly among nodes. Then we construct a circular list of

experimental images and process groups of size N of them in a pipelined fashion. In step one we assign experimental image $I_1$ to processor $P_1$ , $I_2$ to $P_2$, … $I_N$ to $P_N$ and compute the best correlation coefficients, $\sigma_{1,1}$, $\sigma_{2,2}$ , … $\sigma_{N,N}$ . Then in step two $I_1$ migrates to $P_2$ , $I_2$ to $P_3$, … $I_N$ to $P_1$ and compute the best correlation coefficients, $\sigma_{1,2}$ $\sigma_{2,3}$ , … $\sigma_{N,1}$ . After N steps we determine the orientation of experimental image "i" such that $\sigma_i = \min [\sigma_{i,1}, \sigma_{i,2}$ , … $\sigma_{i,,N}$ ] and continue until the entire pool of experimental images is exhausted. The point we make is that the agent needs also to select a particular algorithm based upon the available data, in this case select one algorithm when the symmetry is known and a different one when it is not.

**Conclusions.** While it is conceivable that the members of the lab where the programs have been developed are able to make intelligent choices, *such knowledge tends to degrade in time and space*. If the human user takes the algorithms and methodologies discussed in this section for granted then the results will be unsatisfactory. Either the performance of the program or the quality of the solution or both may be far from optimal. It is unrealistic to expect that a structural biologist will be able to use high performance computing and sophisticated graphics systems without some form of automated knowledge sharing and management.

### C.2.V. WORKFLOWS, EXPERIMENT PLANNING, AND KNOWLEDGE MANAGEMENT.

**Overview. A** combined planning and workflow enactment engine integrated with a knowledge management system is at the heart of the Virtual Laboratory. The function of the planning engine is to create a plan describing a set of actions/tasks necessary to reach various goals. In case of electron microscopy the tasks are: (a) prepare samples, (b) gather experimental data, (c) extract particle projections from micrographs, (d) obtain the orientations of the experimental projections, (e) perform the 3D reconstruction, (f) carry out atomic level modeling, and so on. Some of these tasks are computational e.g. (c), (d), (e), others are not, e.g. (a), (b). It is easy to see that some of the tasks above are complex; they in turn can be decomposed into smaller tasks. The result of planning is what we later describe as a workflow. The *workflow enactment engine* is a computer program that triggers the execution of a task once the necessary resources are available, then, after the task is done, triggers the next one(s). For example extracting the projections of an image may involve data migration from the electron microscope site to the processing site, interaction with the user to get a set of input parameters for the identification algorithm, testes with various combinations of parameters as described in the previous section in Example 1. Sometimes a human carries out the tasks rather than a computer and, in such cases, the workflow enactment engine simply asks the human to provide information when the task was finished, e.g. the settings of a microscope once a sample was mounted.

The main problem is that such workflows are *dynamic*, often a step must be repeated under a different set of conditions, backtracking or even trying a different set of actions may be necessary. For example, in Example 2 of the previous section, the multi-phase algorithm may lead to incorrect results simply because the features of the virus are too fine and they are lost in the global orientation determination phase. This would be discovered at some later point in time e.g., during the atomic modeling, trying to fit known atomic level structures from Protein Data Bases onto the high resolution electron density maps obtained as a result of the 3D reconstruction process. Then the planning algorithm need to re-activated. Eventually the planning engine will create a new workflow based upon a single-step orientation determination. Thus planning and workflow enactment engine must be integrated.

Last, but not least, the system needs a knowledge management system and several ontolgies. For example we need metadata to describe Bond components, individual strategies in the strategy databases, planes, blueprints. A strategy will be described by its input parameters, their types, their range, the output parameters, type and range, a set of preconditions and so on. The metadata describing a blueprint will identify the function of the agent, possibly the subprotocols understood by the agent, and provide links to

the metadata describing the planes and/or the strategies invoked by the blueprint. Another ontology will cover the legacy programs and describe the format of their input and output data, the format of their control input files, and all the internal parameters of the program, the algorithms used and so on. Now we address each of these issues separately.

**A Petri Net Based Workflow Enactment Engine.** A workflow describes a complex activity where individual tasks have to be correlated with one another. For example, the processing of an insurance claim, the activity in a mailroom, and the assembly of an automobile can be described as workflows. Scripts are often used in metacomputing, and can be assimilated with simple workflows, they describe how to execute a group of programs. A workflow abstracts spatial and temporal constraints of a process, without specifying the resources needed for the task. Workflows are very common to describe industrial and business processes. The Workflow Coalition is an organization attempting to provide standards to describe workflows, one of them being the Workflow Definition Language, WDL. We have already implemented a workflow enactment engine and a translator from WDL into Blueprint [102]. In fact we translate WDL into Petri Nets, PNs, and then PNs into Blueprint. Once the blueprint is generated we can create a Bond agent that will act as a *case manager* for the workflow. If a monitoring agent acting as a beneficiary, detects changes of the environment then it triggers agent surgery and a new agent/case manager is generated. We proved that well-formed workflow descriptions lead to free-choice PNs and that there is an isomorphism between such nets and multi-plane agents. We describe an algorithm for S-decomposition used to map a free-choice PN to an agent blueprint in [102]. S-decomposition is the process of decomposing a PN into set of state machines, each state machine supports choice but not concurrent execution.

There are several advantages in using PNs to describe workflows: (1) PNs provide a well researched computational model, there are many tools to analyze PNs and detect potential problems in the definition of an workflow described as a PN, while WDL descriptions may be ambiguous or incorrect, (2) there is a body of work on workflow inheritance based upon PN models. This is very useful to relate workflows derived from the same generic description or for inter-organizational workflows, a very hot subject for industrial applications of workflows. The inter-organizational workflows could be used for collaborative efforts when groups dispersed throughout the globe carry out an experiment. A group led by Wil Van der Aalst is leading this effort and we collaborate with his group. (3) We plan to continue our earlier work on system modeling, [74], [75], and work on modeling agent systems using PNs.

We decided to augment the enactment engine of the agent factory with two more semantic engines one based upon Petri Nets and one on state charts [62]. The state charts semantics may be used for real time applications e.g. the control of the electron microscope. The new PN engine will be integrated with planning and scheduling. Resources will be bound to a task either at the beginning or this binding may be deferred until the task has to be executed next. At that time, the planning engine will evaluate various scenarios based upon a set of scheduling rules and facts. We need to extend the concept of reversibility of a PN to decide what is the optimal marking of the net to be used as the starting state, when we need to carry out re-planning. Another challenge is to ensure system reliability e.g. by creating a shadow case agent, and use the subscription model for monitoring already implemented in Bond to detect the failure of the original case manger.

**Hierarchical Planning.** One of the characteristics of an agent is intelligent behavior and planning is an integral part of this behavior. An agent is a reactive program, it runs through the following steps repeatedly: (a) it generates a goal to achieve, (b) constructs a plan to reach this goal from its current state, and (c) keeps executing this plan until the plan is finished. The agent must have access to a knowledge base where it finds (1) *actions* - generally programs that generate successor state descriptions, (2) *states* - often a state is a data structure, (3) *goals* - generally a function to test if the agent has reached its goal, and (4) *plans* - a plan is a sequence of actions.

A system evolves in time by interacting with its environment trough actions. As a result of these actions it traverses a set of states. Some of the states of a system may be more desirable than others and often we can identify goal states, states a system attempts to reach. Problem solving is the process of establishing a goal and finding sequences of actions to achieve this goal. More specifically problem solving involves three phases: (a) goal formulation, (b) problem formulation, deciding upon a set of actions and states, and (c) execution of the actions. A problem is defined in an environment represented by a state space, and the system evolves from an *initial state*, the state the system finds itself in at the beginning of the process, towards a *goal state* following a path called a solution. Two functions are necessary for problem solving, *a goal test function* to determine when the system has reached its goal state and a *path cost function* to associate a cost with every path. In the general case finding an optimal path from the initial state, to the goal state requires to search a possibly very large space of states. When several paths lead to the goal state we are determined to find the least cost path. Problem solving requires the search of solution spaces with large branching factors and depth of the search tree.

Problem-solving and planning are related but there are several subtle differences between them. First, in problem solving we always consider a sequence of actions starting from the initial state. This makes the problem very difficult, because the number of choices in the initial state is enormous for real-life problems. In planning we can take a more reasonable approach we may work on the part of the problem that most likely to be solvable with the current knowledge. There is no connection between the order of planning and the order of execution.

Planning algorithms use a formal language, usually first-order logic, to describe states, goals, and actions. States and goals are represented by sentences and actions by logical descriptions of their preconditions and effects. Last but not least in planning we can use a divide-and-conquer method to accomplish conjunctive goals. We intend to use backward chaining, partial-order, hierarchical planning algorithms for this project.

**Knowledge Management**. Agents need a language to express the knowledge and the means to carry out reasoning in that language. This language is known as *knowledge representation language*. Here we discuss only frame systems based upon an emerging frame-based knowledge model standard and knowledge base interaction API, called *Open Knowledge Base Connectivity* , OKBC. This standard has been incorporated in Protege, a knowledge base creation and maintenance tool developed at Stanford [3]. The universe of discourse is a set of entities about which knowledge is to be expressed, as well as all constants of the basic types (true, false, integers, floating point numbers, strings, symbols, lists, classes). Classes are sets of entities, and all sets of entities are considered to be classes. A *frame* is a primitive object that represents an entity in the domain of discourse. Formally, a frame corresponds to a KIF constant. A frame has associated with it a set of *own slots*, and each own slot of a frame has associated with it a set of entities called *slot values*. Formally, a slot is a binary relation, and each value **V** of an own slot **S** of a frame **F** represents the assertion that the relation **S** holds for the entity represented by **F** and the entity represented by **V**, i.e., **(S F V)**. An own slot of a frame has associated with it a set of *own facets*, and each own facet of a slot of a frame has associated with it a set of entities called *facet values*. Formally, a facet is a ternary relation, and each value **V** of own facet **Fa** of slot **S** of frame **Fr** represents the assertion that the relation **Fa** holds for the relation **S**, the entity represented by **Fr**, and the entity represented by **V**, i.e., **(Fa S Fr V)**. A *class* is a set of entities. Each of the entities in a class is said to be an *instance* of the class. An entity can be an instance of multiple classes, which are called its *types*. A class can be an instance of a class. A class which has instances that are themselves classes is called a *meta-class*. Entities that are not classes are referred to as individuals. Thus, the domain of discourse consists of individuals and classes. A class frame has associated with it a collection of *template slots* that describe own slot values considered to hold for each instance of the class represented by the frame. The values of template slots are said to inherit to the subclasses and to the instances of a class. Formally, each value **V** of a template slot **S** of a class frame **C** represents the assertion that the relation template-slot-value holds for the relation **S**, the class represented by **C**, and the entity represented by **V** , i.e. the template-slot-value (**S C V)**. That assertion, in turn, implies that the relation **S** holds between each instance **I** of class **C** and value **V** i.e., (**S I V)**. It also implies that the relation template-slot-value holds for

the relation **S**, each subclass **Csub** of class **C**, and the entity represented by **V**, i.e.,(**template-slot-value S Csub V**). *A knowledge base, KB* is a collection of classes, individuals, frames, slots, slot values, facets, facet values, frame-slot associations, and frame-slot-facet associations. KBs are considered to be entities in the universe of discourse and are represented by frames. All frames reside in some KB. The knowledge representation mode of OKBC is considerably richer that that of a programming language like Java, it supports multiple inheritance, definition of classes of classes, the runtime checking of the type and range of values of an item.

Protege 2000 [3] is an integrated software tool used by system developers and domain experts to develop knowledge-based systems. The current version is implemented in Java and employs the OKBC knowledge model. Protege has three layers (1) the knowledge base server, a wrapper around an actual knowledge base server, (2) the control layer that handles standard actions and connection between widgets and underlying knowledge base, and (3) a widget layer consisting of user interface components that allow a small slice of the knowledge base to be viewed and edited. Protege is able to store the metadata either as facts compatible with the Clips and to some extent to Jess or in RDF format. *The Resource Description Framework, RDF*, is a World Wide Web Consortium standard for metadata on the Internet. The syntax of RDF is defined in terms of XML and its data model is essentially that of semantic nets or frames, with a few important extensions. An RDF description can be created about any resource that can be referred to using an Uniform Resource Identifier (URI). Such a description is itself stored in a resource and can be referred to using an URI and thus described by another RDF description. An RDF description is a set of triples **(A u1 u2**) where **A** is the assertion identifier determining the property whose subject is described by URI **u1** and value by URI **u2**. Such sets can be easily aggregated unlike arbitrary XML documents.

We are in the process of integrating Protege with Bond. Protege is available as an open source thus we will be able to develop the code to map Bond objects into metadata and metadata into Bond objects. The main advantage of this approach is that graphical user interfaces for defining metadata are already provided thus we will be able to start building knowledge bases soon. We might need to collaborate with the Stanford group and with the group at Sandia to make sure the format generated by Protege is compatible with Jess.

### C.2.VI. MILESTONES.

**Year 1**.

- (Y1.1) Develop a Petri Net based workflow enactment engine.
- (Y1.2) Integrate the Protege knowledge management system in Bond. Make sure that the output generated by Protege is compatible with Jess, the inference engine used by Bond.
- (Y1.3) Provide an XML version of the agent description language, the blueprint.
- (Y1.4) Extend the blueprint to generate federation of agents, tightly coupled agents. Evaluate the use of conversations for inter-agent communication as suggested by Bradshaw.
- (Y1.5) Work on interoperability with other systems. Integrate Bond and Jini. Allow Bond to access Jini services. Evaluate the advantages of integration of Bond with the E-Speak system from HP.
- (Y1.6) Using Protege, develop the metadata necessary for composition of Bond strategies and legacy programs.

**Year 2**.

- (Y2.1) Develop and implement hierarchical planning algorithms. The planning engine should be implemented as a strategy, exactly like the inference strategy based upon Jess.
- (Y2.2) Integrate planning and the workflow enactment engines to support re-planning. In other words support some form of backtracking using the reversibility property of Petri Nets.
- (Y2.3) Develop a framework for negotiations and broker agents.
- (Y2.4) Investigate bidding schemes for resource management.

- (Y2.5) Start building the domain knowledge bases for 3D Electron microscopy. Start with knowledge based for describing programs and data for 3D reconstruction and plans for data processing.

**Year 3**.

- (Y3.1) Develop an efficient scheme for Web based agent control using the microserver concept implemented in the current version of Bond.
- (Y3.2) Develop the agent for controlling the automatic particle identification.
- (Y3.3) Develop the agent for parallel orientation and 3D reconstruction.
- (Y3.4) Develop a resource management scheme based upon agents.
- (Y3.5) Continue building the domain knowledge bases for 3D Electron microscopy with those related the control of the electron microscope and sample preparation.

**Year 4.**

- (Y4.1) Design benchmarks for the Virtual Laboratory. Obtain quantitative performance measures.
- (Y4.2) Optimize critical system functions.
- (Y4.3) Study scalability issues.
- (Y4.4) Develop qualitative criteria for comparing agent-based, distributed object systems.
- (Y4.5) Develop new applications.
- (Y4.6) Continue building domain knowledge bases.
- (Y4.7) Document the system.

## C.2.VII IMPACT ON EDUCATION.

A main trust of this proposal is its support for education. A first year graduate student or postdoctoral fellow in a structural biology lab typically relies on folklore and nearly incomprehensible descriptions of programs or experimental procedures. It is not unusual that in a lab of 30 to 40 people there is a single person who knows all critical details for a successful experiment and a novice would have to wait for months to understand why the results produces by a program are obviously wrong. Even worse, she may find out after years of work that she had used a wrong set of model parameters in an early stage of her data analysis. The Virtual Laboratory will create an infrastructure to facilitate sharing of knowledge. A novice would replay an earlier experiment and examine the set of model parameters or the exact sequence of steps necessary to prepare a sample or to run a program, and then she may contribute herself by adding to the knowledge base her comments. A word of caution, the Virtual Laboratory will only be as helpful as the knowledge stored by its databases. Our collaborations with the groups of Michael G. Rossmann and Timothy Baker, world renown structural biologists and co-PI of other joint research projects, guarantee that (1) the knowledge collected will be accurate, (2) there are plenty of graduate students and Post Doctoral Fellows to use the system and to contribute to it. Though it seems far-fetched at this time, the Virtual Laboratory could eventually propagate to high schools and help students learn the basic facts about scientific discovery in biology.

The computer science aspect of this project. Three graduate students are working towards their Ph.D. on subjects related to distributed object systems and agent-based computing. In Fall 98, we offered a graduate level seminar on Network-Centric Computing for students and faculty in the CS and ECE Departments at Purdue used Bond for the projects. One of these projects was the basis for the network of PDE solvers mentioned earlier. The graduate level course in Distributed Computing the PI taught during the Spring 2000 semester used Bond for various projects. Several groups that have downloaded Bond have listed educational use as the intended use of the system.

## D. LITERATURE (Alphabetical)


1. Atallah, M.J., C. Lock, D.C. Marinescu, T.L. Casavant, and H.J. Siegel (1992). Models and algorithms for co-scheduling compute-intensive tasks on a network of workstations. *J. Paral. & Distrib. Comput.*, **16(4):319–327**.
2. Atwood, J.W., O. Catrina, J. Fenton, and W.T. Strayer (1996). Reliable multicasting in the Xpress Transport protocol. In *Proc. 21st Local Computer Networks Conf.*, **202–211**.
3. Baldonado, M., C. Chang, L. Gravano, and A. Paepcke (1997). The Stanford digital library metadata architecture. *Int'l. J. Digital Libraries*, **1(2):108–121**.
4. Barzilai, T., D. Kandlur, A. Mehra, D. Saha, and S. Wise (1998). Design and implementation of an RSVP based quality of service architecture for 'Integrated Services Internet'. *IEEE J. Selected Areas in Communications*, **16(4)**.
5. Bäumer, C., M. Breugst, S. Choy, and T. Magedanz (1999). Grasshopper — a universal agent platform based on OMG MASIF and FIPA standards. In *First International Workshop on Mobile Agents for Telecommunication Applications* (*MATA'99*) (A. Karmouch and R. Impley, eds.), World Scientific Publishing Ltd., Ottawa, Canada, **1–18**.
6. Berman F. (1998). High performance schedulers. In *The Grid. Blueprint for a New Comput. Infrastructure* (I. Foster and C. Kesselman, eds.). Morgan Kaufmann**, 279–311**.
7. Bhatia, D., V. Burzevski, M. Camuseva, G. Fox, W. Furmanski, and G. Premchandran (1997). WebFlow — A visual programming paradigm for Web/Java based coarse grain distributed computing. *Concurrency: Practice and Experience*, **9(6):555–578**.
8. Bölöni L., R. Hao, K.K. Jun, and D.C. Marinescu (1998). Bond: a distributed object system. ISCOPE 98, http://bond.cs.purdue.edu.
9. Bölöni L. and D.C. Marinescu (1999). On the robustness of metaprogram schedules. *Proc Heterogeneous Computing Workshop, HCW'99*, IEEE Press, **146–155**.
10. Bölöni L. and D.C. Marinescu (1999). Three theorems on robustness of metaprogram schedules. *Proc. ACM ICS Workshop on Scheduling Algorithms*, Rhodes, IEEE Press (in press).
11. Bölöni L., R. Hao, K.K. Jun, and D.C. Marinescu (1999). Structural biology metaphors applied to the design of a distributed object system. In *Proc. Second Workshop on Bio-Inspired Solutions to Parallel Processing Problems*, LNCS, Vol. 1586, Springer Verlag, **275–283**.
12. Bölöni L., D.C. Marinescu, J.R. Rice, P. Tsompanopoulou, and M. Vavalis (2000). Agent based networks for scientific simulation and modeling. *Concurrency: Practice and Experience*, 2000 (in press).
13. Bölöni L. and D.C. Marinescu (2000). An object-oriented framework for building collaborative network agents. In *Intelligent Systems and Interfaces* (N.H. Teodorescu , D. Mlynek, A. Kandel, H.J. Zimmerman, eds.). Kluewer Publishing House, **31–65**.
14. Bölöni L. and D.C. Marinescu (2000). Robust scheduling of metaprograms. *Journal of Scheduling*, Willey, 2000 (in press).
15. Bölöni L. and D.C. Marinescu (2000). A component-based agent model – from theory to implementation. In *Proc. Second Int. Workshop "From Theory to Agent Implementation", Vienna*, April 2000 (in press).
16. Bölöni L. and D.C. Marinescu (2000). A multi-plane agent model. *In Proc. Autonomous Agents 2000*, Barcelona, June 2000 (in press).
17. Bölöni L., K.K. Jun, K. Palacz, R. Sion and D.C. Marinescu (2000). The Bond agent system and Applications. In *Proc. ASA/AM 2000*, LNCS, Springer Verlag, 2000 (in press)
18. Bölöni L. and D.C. Marinescu (2000). Agent surgery: the case for mutable agents. In *Proc. Third Workshop on Bio-Inspired Solutions to Parallel Processing Problems*, LNCS,

Springer Verlag (in press).

19. Bradshaw, J.M., S. Dutfield, P. Benoit, and J.D. Woolley (1997). KaoS: toward an industrial-strength open agent architecture. In *Software Agents*. MIT Press, **375–418**.
20. Bradshaw, J.M. (1997). An introduction to software agents. In *Software Agents*. MIT Press, **5–46**.
21. Breugst, M., I. Busse, S. Covaci, and T. Magedanz (1998). Grasshopper — a mobile agent platform for IN based service environments. In *Proceedings of IEEE IN Workshop 1998*, Bordeaux, France, **279–290**.
22. Busse, I., B. Deffner, and H. Schulzrinne (1996). Dynamic QoS control of multimedia applications based on RTP. *Computer Communications*, **19(1):49–58**.
23. Busetta, P. R. Rnnquist, A. Hodgson, and A. Lucas. JACK intelligent agents — components for intelligent agents in java.
    `URL http://agent-software.com.au/whitepaper/html/index.html`.
24. Caglayan A. and C. Harrison (1997). Agent sourcebook. Willey.
25. Camiel, N., S. London, N. Nisan, and O. Regev (1997). The POPCORN project: distributed computation over the internet in Java. In *Proc. Sixth Int'l. World Wide Web Conf.*
26. Catlett, C. and L. Smarr (1992). Metacomputing. *Communications of the ACM*, **35(6):44–52**.
27. Chandy, M.K., A. Rifkin, P. Silviotti, J. Mandelson, M. Richardson, W. Tanaka, and L. Weissman (1996). A world-wide distributed system using Java and the internet. In *Proc. IEEE Int. Symp. of High Performance Distributed Comput*.
28. Chun, B., A. Mainwaring, and D. Culler (1997). Virtual network transport protocols for Myrinet. In *Proc. Hot Interconnects V.*, Los Alamitos, CA. IEEE Computer Society Press.
29. Cohen P.R. et. al. (1998). An open agent architecture. In *Readings in Agents* (M.N. Huhns and M.P. Singh, eds.). Morgan Kaufmann publishers, **185–196**.
30. Connelly, K. and A.A. Chien (1997). FM-QoS: Real-time communication using self-synchronizing schedules. In *Proc. SuperComputer'97*.
31. Costian, C., I.M. Martin, Cornea-Hasegan, M.A., D.C. Marinescu, and J.R. Rice (1994). Towards problem solving environments in high performance computing. In P*roc. High Performance Comput. Conf.'94*, **354–366**.
32. Costian, C. and D.C. Marinescu (1995). SOCRATES: an environment for high performance computing. In *Proc. 4th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE Press.
33. Cousins, S., H. Garcia-Molina, S. Hassan, S. Ketchpel, M. Roscheisen, and T. Winograd (1996). Towards interoperability in digital libraries. *IEEE Computer*, **29(5)**.
34. Cover, T.M. and J.A. Thomas (1991). Elements of information theory. Willey.
35. Daniel, R. and C. Lagoze (1997). Distributed active relationships in the Warwick framework. In *IEEE Metadata Conf.*, Los Alamitos, CA. IEEE Computer Society Press.
36. Dincer, K. and G. Fox (1997). Using Java and JavaScript in the virtual programming laboratory: a web-based parallel programming environment. *Concurrency: Practice and Experience*, **9(6):521–534**.
37. Dolev, D. and D. Malki (1996). The Tranis approach to high availability cluster communication. *Communications of the ACM*, **39(4):64–70**.
38. Eriksson, H. (1994) Mbone: The multicast backbone. *Communications of the ACM*, **37(4):50–54**.
39. Fabre J.C. and T. Perennou (1998). A meta-object architecture for fault-tolerant distributed systems. *IEEE Trans. on Computers*, **47(1):78–95**.
40. Ferguson, P. and G. Huston (1998). *Quality of service: Delivering QoS in the internet and*

*the corporate network*, New York. Wiley Computer Books.

41. Finin T. and al. (1993). Specification of the KQML agent-communication language. *DARPA Knowledge Sharing Initiative* (draft).
42. Finin T, Y. Labrou, and J. Mayfield (1997). KQML as an agent communication language. In *Software Agents* (J.M. Bradshaw, ed.). MIT Press, **291–316.**
43. FIPA Specifications. `URL http://www.fipa.org`.
44. Foster, I., J. Geisler, C. Kesselman, and S. Tuecke (1997). Managing multiple communication methods in high-performance networked computing systems. *J. Paral. and Distr. Comput*., **40:35–48**.
45. Foster, I. and C. Kesselman (1997). Globus: A metacomputing infrastructure toolkit. *Int'l. J. Supercomputing Applications*, **11(2):115–128**.
46. Foster, I. and C. Kesselman (1998). Computational grids. In *The Grid. Blueprint for a New Comput. Infrastructure* (I. Foster and C. Kesselman, eds.), Morgan Kaufmann, **15–52**.
47. Fox, G. and W. Furmanski (1997). Petaops and exaops: Supercomputing on the Web. *IEEE Internet Comput*., **1(2):38–46**.
48. Franklin, S. and A. Graessner (1996). Is it an agent, or just a program? In *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*. Springer Verlag.
49. Friedman-Hill, E. (1999). Jess, the java expert system shell. Technical Report SAND98-8206, Sandia National Laboratories.
50. Gamma, E., R. Helm, R. Johnson, and J. Vlissides (1997). Design patterns: elements of reusable object-oriented software. Addison Wesley.
51. Geist, G.A., J.A. Kohl, and P.M. Papadopoulos (1997). CUMULVS: Providing fault-tolerance, visualization, and steering of parallel applications. *Int'l. J. Supercomputer Appl.*, **11(3):224–236**.
52. Generserth, M.R. and R.E. Fikes, et al. (1992). Knowledge interchange format. *Version 3.0 reference manual*. Technical Report Logic-92-1, Computer Science Department, Stanford University.
53. Genesereth M.R. (1997). An agent-based framework for interoperability. In *Software Agents* (J.M. Bradshaw, ed.), MIT Press, **317–347**.
54. Georgeff, M., D. Kinney, and A. Rao (1966). A methodology and modeling technique for systems of BDI agents. In *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modeling Autonomous Agents in a Multi-Agent World* (LNAI Volume 1038) (W. Van de Velde and J.W. Perram, eds.), Springer-Verlag, **page 56**.
55. Gokhale, A. and D. Schmidt (1997). Evaluating CORBA latency and scalability over high-speed ATM networks. In *Proc. IEEE 17th Int'l. Conf. on Distributed Systems*. Los Alamitos, CA. IEEE Computer Society Press.
56. Gosling, J., B. Joy, and G. Steele (1996). *The Java language specification*. Addison-Wesley.
57. Greaves, M., H. Holmback, and J.M. Bradshaw (1999). What is a conversation policy? In *Proceedings of the Autonomous Agents'99 Workshop on Specifying and Implementing Conversation Policies*, Seattle, WA.
58. Grimshaw A. and W. Wulf (1996). Legion – a view from 50,000 feet. In *Proc. IEEE Int. Symp. of High Performance Distributed Comput*. IEEE Press, **89–99**.
59. Guerin R. and H. Schulzerinne (1998). Network quality of service. In *The Grid. Blueprint for a New Comput. Infrastructure* (I. Foster and C. Kesselman, eds.), Morgan Kaufmann Publishers, **479–504**.
60. Hao R., L. Bölöni, K.K. Jun and D.C. Marinescu (1998). Bond system security and access control models. *Proc IASTED Conference on Parallel and Distributed Computer and*

*Networks*, **520–524**.

61. Hao R., L. Bölöni, K.K. Jun and D.C. Marinescu (1999). An aspect-oriented approach to distributed object security. *Proc. 4-th IEEE Symp. on Computers and Communication*, IEEE Press (in print).
62. Harel, D. A. Pnueli, J.P. Schmidt, and R. Sherman (1987). On the formal semantics of statecharts. In *2nd IEE Symposium on Logic in Computer Science*.
63. Hertmann, P. (1997). *Illustrated guide to HTTP*. Manning Publications.
64. Johnson W.E. (1998). Real-time widely distributed instrumentation systems. In *The Grid. Blueprint for a New Comput. Infrastructure* (I. Foster and C. Kesselman, eds.), Morgan Kaufmann Publishers, **74–104**.
65. Jun K.K., L. Bölöni, D.K.Y. Yau, and D.C. Marinescu (2000). Intelligent QoS support for an adaptive video service. *Proc. IRMA 2000, International Resource Management Association* (in press).
66. Jun K.K., L. Bölöni, K. Palacz, and D.C. Marinescu (2000). Agent based resource discovery. *Proc Heterogeneous Computing Workshop, HCW'00*, IEEE Press (in press).
67. Kelly, F. (1996). Notes on effective bandwidth. *Royal Statistical Society Lecture Notes*, **4:141–168**.
68. Kieffer, J. (1995). Prediction and information theory. Six notes on information theory. http//www.ece.umn/users/kieffer.
69. Knapik M. and J. Johnson (1998). *Developing Intelligent Agents for Distributed Systems*. McGraw-Hill.
70. Knudsen J. (1998). Java cryptography. O'Reilly.
71. Kunchithapadam, K. and B.P. Miller (1996). Integrating a debugger and a performance tool for steering. In *Debugging and Performance Tools for Paral. Comput. Systems* (M.L. Simmons, A.H. Hayes, J.S. Brown, and D.A. Reed, eds.). Los Alamitos, CA. IEEE Society Press, **53–64**.
72. Labrou, Y., T. Finn (1997). A proposal for a new KQML specification. *UMBC TR-CS-97-03*.
73. Leigh, J., A. Johnson, and T.A. DeFanti (1997). CAVERN: A distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments. *Virtual Reality: Research, Development and Applications***, 2(2):217–237**.
74. Lin C. and D.C. Marinescu (1988). High level Petri nets and applications. *IEEE Transactions on Computers*, **36(7):815–826**.
75. Lin C., A.B. Whinston, A. Chaudhuri, and D.C. Marinescu (1993). Logical inference of Horn clauses in Petri net models. *IEEE Transactions on Knowledge Engineering*, **5(3):815–826**.
76. Lopez C. and G. Kiczales (1998). Aspect-Oriented programming with aspect J. http:www.parc.xerox.com/aop.
77. Lynch R.E., D.C. Marinescu, H. Lin, and T.S.Baker (1999). Parallel algorithms for reconstruction of asymmetric objects from electron micrographs. *Proc. IPPS 99*, IEEE Press, **632-637**.
78. Lynch R.E., H. Lin, and D. C. Marinescu (2000). An algorithm for 3D Reconstruction of Asymmetric Objects. *SIAM Journal of Scientific Computing* (submitted).
79. Lynch R.E., H. Lin, and D. C. Marinescu (2000). An efficient algorithm for parallel 3D reconstruction of asymmetric objects, from electron micrographs, *Proceedings of Euro-Par 2000*, LNCS, Springer Verlag 2000 (to appear).
80. Marinescu, D.C, J.E. Lump, T.L. Casavant, and H.J. Siegel (1992). Models for monitoring and debugging tools for parallel and distributed software. *J. Paral. & Distrib. Comput.*, **9(2):171–84**.

81. Marinescu, D.C. and J.R. Rice (1994). On high-level characterization of parallelism. *J. Paral. & Distrib. Comput*., **20:107–113**.

82. Marinescu, D.C. and J.R. Rice (1994). On scalability of asynchronous parallel computations. J. *Paral. & Distrib. Comput*., **22:538–546**.

83. Marinescu, D.C, L. Bölöni, R. Hao, and K.K. Jun(1998). An Alternative model for scheduling on a computational grid. In *Proc. 13-th Int. Symp. On Computer and Information Sciences, IOS Press,* **473–481.**

84. Marinescu, D.C. (1998). Software development for intranet applications. In *Proc. IFIP Workshop on Dependable Comput. and its Applications*. DCIA'98, **31–50**.

85. Marinescu, D.C. and L. Bölöni.(1999). Biological metaphors applied to the design of complex software systems. *Journal of Future Computer Systems*. Elsevier (in press).

86. Marinescu, D.C. (1999). An agent-based design for Problem Solving Environments. *Proc. Workshop on Parallel/High Performance Scientific Computing, POOSC'99* (in press).

87. Marinescu, D.C. and L. Bölöni.(1999). A component architecture for Problem Solving Environments. *Proc. Symp. on Computational Science* (in press).

88. Martin, I.M., D.C. Marinescu, and J.R. Rice (1995). Adaptive load balancing strategies for solving irregular problem on distributed memory MIMD systems. In *Proc. IPPS 95*. IEEE Press, **57–64**.

89. Martin I.M. and D.C. Marinescu (1996). Exploiting symmetry in parallel computations for structural biology. In *Proc. Euro-Par'96, European Paral. Proc. Conf.* LNCS, Springer Verlag, **1124:255–259**.

90. Martin, I.M. and D.C. Marinescu (1998). Concurrent computations and data visualization for structure determination of spherical viruses. *IEEE Computational Science and Engineering* (in press).

91. Martin, I. M., D. C. Marinescu, R. E. Lynch., and T. S. Baker (1997). Identification of spherical virus particles in digitized images of electron micrographs. *J. Struct. Biol.* **120, 146-157**.

92. MASIF - The CORBA mobile agent specification.
URL `http://www.omg.org/cgi-bin/doc?orbos/98-03-09`.

93. Messina P. (1998). Distributed supercomputer applications. In *The Grid. Blueprint for a New Comput. Infrastructure* (I. Foster and C. Kesselman, eds.). Morgan Kaufmann Publishers, **55–74**.

94. Minsky, M.L. (1975). A framework for representing knowledge. In *The Psychology of Computer Vision* (P.H. Winston, ed.). McGraw-Hill, New York, **211–277**.

95. Ndumu, D. H. Nwana, L. Lee, and H. Haynes (1999). Visualization of distributed multi-agent systems. *Applied Artificial Intelligence*, **13(1):187-208**.

96. Neuman, B.C. and S. Rao (1994). The Prospero Resource Manager: A scalable framework for processor allocation in distributed systems. *Concurrency: Practice and Experience*, **6(4):339–355**.

97. Neuman, B.C. and T. Ts'o (1994). Kerbero: An authentication service for computer networks. *IEEE Communications*, **32(9):33–39**.

98. Nieh, J. and M. Lam (1997). The design, implementation and evaluation of SMART: A scheduler for multimedia applications. In *Proc. 16th Symp. on Operating System Principles*, **184–197**.

99. Nwana, H. and D. Ndumu. (1999). A perspective on software agents research. *The knowledge Engineering Review*.

100. Nwana, H. D. Ndumu, L. Lee, and J. Collis (1999). Zeus: a toolkit for building distributed multi-agent systems. *Applied Artificial Intelligence*, **13 (1):129–186**.

101. Orfali, R. and D. Harkey (1997). *Client/server programming with Java and CORBA*. New York. John Wiley and Sons.
102. Palacz K and D.C. Marinescu (2000). *An agent-based workflow management system.* In *Proc. Workshop Bringing Knowledge to the Business Process*, Stanford University, AAAI Press, **119–127**.
103. Paolucci, M. D. Kalp, A. Pannu, O. Shehory, and K. Sycara (1998). *Lecture Notes in Artificial Intelligence, Intelligent Agents, Chapter A: Planning Component for RETSINA Agents*. Springer-Verlag, Heidelberg.
104. Patil R.S. et. al. (1998) The Darpa knowledge sharing effort: progress report (M.N. Huhns and M.P. Singh, eds.). Morgan Kaufmann Publishers, **243–254**.
105. Petrie, C. (1996). Agent-based engineering, the web, and intelligence. *IEEE Expert*, **11(6):24–29**.
106. Postel J. and J. Touch (1998). Network infrastructure. In *The Grid. Blueprint for a New Comput. Infrastructure* (I. Foster and C. Kesselman, eds.), Morgan Kaufmann, **533–566**.
107. Potter, C., R. Brady, P. Moran, C. Gregory, B. Carragher, N. Kisseberth, J. Lyding, and J. Lindquist (1996). EVAC: A virtual environment for control of remote imaging instrumentation. *IEEE Computer Graphics and Applications*, **62–66**.
108. Rao, A.S. and M.P. Georgeff, (1995). BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems* (V. Lesser, ed.), San Francisco, CA MIT, **312–319**.
109. Rao, A.S. and M.P. Georgeff (1999). Modeling rational agents within a BDI-architecture. In *Proceedings of Knowledge Representation and Reasoning* (*KR&R-91*), **473–484**.
110. Reed, D.A., R.C. Giles, and C.E. Catlett (1997). Distributed data and immersive collaboration. *Communications of the ACM*, **40(11):39–48**.
111. Russell, S.J. and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.
112. Schmidt, D.C. (1997). Lessons learned building reusable OO frameworks for distributed software. *Communications of the ACM*, **40(10)**.
113. Schmidt, D.C., A. Gokhale, T. Harrison, and G. Parulkar (1997). Towards real-time CORBA. *IEEE Communications Magazine*, **35(2):72–77**.
114. Schulzrinne, H., S. Casner, R. Frederick, and V. Jacobson (1996). RTP: A transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force.
115. Sessions, R. (1998). *COM and DCOM: Microsoft's vision for distributed objects*. New York. John Wiley and Sons.
116. Shoham Y. (1997). An overview of agent-oriented programming. In *Software Agents* (J.M. Bradshaw, ed.). MIT Press, **271–290.**
117. Sirbu, M.G. and D.C. Marinescu (1996). Bond – a parallel virtual environment. In *Proc. HPCN 96, High Performance Comput. and Networking*. LNCS, Springer Verlag, Vol. 1067**,** **255–59**.
118. Sirbu, M.G. and D.C. Marinescu (1997). A scheduling expert advisor for heterogeneous environments. In *Proc. HCW'97, Heterogeneous Comput. Workshop*. IEEE Press, **74–83**.
119. Sirbu, M.G. (1997). The design of a meta-computing environment. *Ph.D. Thesis*, Purdue University.
120. Smith, I.A., P.R. Cohen, J.M. Bradshaw, M. Greaves, and H. Holmback (1998). Designing conversation policies using joint intention theory. In *Proceedings of International Joint Conference on Multi-Agent Systems* (*ICMS-98*).
121. Stevens, R., P. Woodward, T. DeFanti, and C. Catlett (1997). From the I-WAY to the national technology grid. *Communications of the ACM*, **40(11):50–61**.

122. Stroud, R.J. (1993). Transparency and reflections in distributed systems. *ACM Operating Systems*, **22(2): 99–103**.

123. Tsompanopoulou, P., L. Bölöni, D.C. Marinescu, and J.R. Rice, (1999). The design of software agents for a network of PDE solvers. In *Proc. Workshop on Agent Technologies for High Performance Computing, Agents 99*, **57–68**.

124. Vahdat, A., P. Eastham, and T. Anderson (1996). WebFS: A global cache coherent file system. *Tech. Report*, Department of Computer Science, UC Berkeley.

125. Vahdat, A., P. Eastham, C. Yoshikawa, E. Belani, T. Anderson, D. Culler, and M. Dahlin (1997). WebOS: Operating system services for wide area applications. *Tech. Report UCB, CSD-97-938*, UC Berkeley.

126. van Steen, M., P. Homburg, L. van Doorn, A. Tannenbaum, and W. de Jonge (1995). Towards object-based wide area distributed systems. In *Proc. Int'l. Workshop on Object Orientation in Operating Systems*, **224–227**.

127. Wang, K.Y., D.C. Marinescu, and O.F. Carbunar (1998). Dynamic scheduling of process groups. *Concurrency: Pract. & Exper*., **10(4):265–283**.

128. White J. (1997). Mobile agents. In *Software Agents* (J.M. Bradshaw, ed.). MIT Press, **437–474.**

129. White, J.E. (1996). Telescript technology: mobile agents. In *Software Agents* (J. Bradshaw, ed.). AAAI/Press/MIT Press.

130. Widerhold G. Mediators in the Architecture of Future Information Systems (1998). In *Readings in Agents* (M.N. Huhns and M.P. Singh, eds.). Morgan Kaufmann Publishers **185–196**.

131. Wilmot, P., S. Howison, and J. Dewynne (1995). The mathematics of financial derivatives. Cambridge University Press.

132. Wooldridge M. and N.R. Jennings (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):**115–152**.

133. Wooldridge, M. N.R. Jennings, and D. Kinney (1999). A methodology for agent-oriented analysis and design. In *Proceedings of the Third Annual Conference on Autonomous Agents* (*AGENTS-99*) (O. Etzioni, J.P. Muller, and J.M. Bradshaw, eds.), New York, ACM Press, **69–76**, May 1-5.

134. Young, S.J., G.G.Y. Fan, D. Hessler, S. Lamont, T.T. Elvins, M. Hadida-Hassan, G.A. Hanyzewski, J.W. Durkin, P. Hubbard, G. Kindlmann, E. Wong, D. Greenberg, S. Karin, and M.H. Ellisman (1996). Implementing a collaborator for microscopic digital anatomy. *Int'l. J. Supercomputer Applications and High Performance Comput*., **10(2/3):170–181**.

135. Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala (1993). RSVP: A new resource reservation protocol. *IEEE Network*, **7(5):8–18**.

136. D.K.Y. Yau , Jun K.K., and D.C. Marinescu (1999) Middleware QoS agents and native kernel schedulers for adaptive multimedia services and cluster servers. *Proc. Real Time Systems Symp* (in press).

# SUMMARY PROPOSAL BUDGET

**YEAR 1**

| FOR NSF USE ONLY | | |
|---|---|---|
| PROPOSAL NO. | DURATION (months) Proposed | Granted |
| AWARD NO. | | |

ORGANIZATION: **Purdue Research Foundation**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Dan C Marinescu**

| A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets) | NSF Funded Person-mos. CAL | ACAD | SUMR | Funds Requested By proposer | Funds granted by NSF (if different) |
|---|---|---|---|---|---|
| 1. **Dan C Marinescu - none** | **0.00** | **0.00** | **0.00** | $ **0** | $ |
| 2. | | | | | |
| 3. | | | | | |
| 4. | | | | | |
| 5. | | | | | |
| 6. ( **0** ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE) | **0.00** | **0.00** | **0.00** | **0** | |
| 7. ( **1** ) TOTAL SENIOR PERSONNEL (1 - 6) | **0.00** | **0.00** | **0.00** | **0** | |
| B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS) | | | | | |
| 1. ( **2** ) POST DOCTORAL ASSOCIATES | **12.00** | **0.00** | **0.00** | **81,600** | |
| 2. ( **0** ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.) | **0.00** | **0.00** | **0.00** | **0** | |
| 3. ( **4** ) GRADUATE STUDENTS | | | | **31,611** | |
| 4. ( **0** ) UNDERGRADUATE STUDENTS | | | | **0** | |
| 5. ( **0** ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY) | | | | **0** | |
| 6. ( **0** ) OTHER | | | | **0** | |
| TOTAL SALARIES AND WAGES (A + B) | | | | **113,211** | |
| C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS) | | | | **21,337** | |
| TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C) | | | | **134,548** | |
| D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) **workstations** $ **16,000** | | | | | |
| TOTAL EQUIPMENT | | | | **16,000** | |
| E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS) | | | | **4,000** | |
| 2. FOREIGN | | | | **0** | |
| F. PARTICIPANT SUPPORT COSTS | | | | | |
| 1. STIPENDS $ **0** | | | | | |
| 2. TRAVEL **0** | | | | | |
| 3. SUBSISTENCE **0** | | | | | |
| 4. OTHER **0** | | | | | |
| TOTAL NUMBER OF PARTICIPANTS ( **0** ) TOTAL PARTICIPANT COSTS | | | | **0** | |
| G. OTHER DIRECT COSTS | | | | | |
| 1. MATERIALS AND SUPPLIES | | | | **0** | |
| 2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION | | | | **1,000** | |
| 3. CONSULTANT SERVICES | | | | **0** | |
| 4. COMPUTER SERVICES | | | | **3,545** | |
| 5. SUBAWARDS | | | | **0** | |
| 6. OTHER | | | | **16,072** | |
| TOTAL OTHER DIRECT COSTS | | | | **20,617** | |
| H. TOTAL DIRECT COSTS (A THROUGH G) | | | | **175,165** | |
| I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **74930 (Rate: 52.0000, Base: 144094)** | | | | | |
| TOTAL INDIRECT COSTS (F&A) | | | | **74,928** | |
| J. TOTAL DIRECT AND INDIRECT COSTS (H + I) | | | | **250,093** | |
| K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.) | | | | **0** | |
| L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K) | | | | $ **250,093** | $ |

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

PI / PD TYPED NAME & SIGNATURE*: **Dan C Marinescu** DATE

ORG. REP. TYPED NAME & SIGNATURE* DATE

| FOR NSF USE ONLY | | |
|---|---|---|
| INDIRECT COST RATE VERIFICATION | | |
| Date Checked | Date Of Rate Sheet | Initials - ORG |
| | | |

# SUMMARY PROPOSAL BUDGET YEAR 2

| FOR NSF USE ONLY | | |
|---|---|---|
| PROPOSAL NO. | DURATION (months) Proposed | Granted |
| AWARD NO. | | |

ORGANIZATION
**Purdue Research Foundation**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR
**Dan C Marinescu**

| A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets) | NSF Funded Person-mos. CAL | ACAD | SUMR | Funds Requested By proposer | Funds granted by NSF (if different) |
|---|---|---|---|---|---|
| 1. **Dan C Marinescu - none** | **0.00** | **0.00** | **1.00** | $ **9,867** | $ |
| 2. | | | | | |
| 3. | | | | | |
| 4. | | | | | |
| 5. | | | | | |
| 6. ( **0** ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE) | **0.00** | **0.00** | **0.00** | **0** | |
| 7. ( **1** ) TOTAL SENIOR PERSONNEL (1 - 6) | **0.00** | **0.00** | **1.00** | **9,867** | |
| B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS) | | | | | |
| 1. ( **2** ) POST DOCTORAL ASSOCIATES | **12.00** | **0.00** | **0.00** | **84,864** | |
| 2. ( **0** ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.) | **0.00** | **0.00** | **0.00** | **0** | |
| 3. ( **3** ) GRADUATE STUDENTS | | | | **24,894** | |
| 4. ( **0** ) UNDERGRADUATE STUDENTS | | | | **0** | |
| 5. ( **0** ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY) | | | | **0** | |
| 6. ( **0** ) OTHER | | | | **0** | |
| TOTAL SALARIES AND WAGES (A + B) | | | | **119,625** | |
| C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS) | | | | **24,319** | |
| TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C) | | | | **143,944** | |
| D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) | | | | | |
| TOTAL EQUIPMENT | | | | **0** | |
| E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS) | | | | **4,000** | |
| 2. FOREIGN | | | | **0** | |
| F. PARTICIPANT SUPPORT COSTS | | | | | |
| 1. STIPENDS $ **0** | | | | | |
| 2. TRAVEL **0** | | | | | |
| 3. SUBSISTENCE **0** | | | | | |
| 4. OTHER **0** | | | | | |
| TOTAL NUMBER OF PARTICIPANTS ( **0** ) TOTAL PARTICIPANT COSTS | | | | **0** | |
| G. OTHER DIRECT COSTS | | | | | |
| 1. MATERIALS AND SUPPLIES | | | | **0** | |
| 2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION | | | | **1,000** | |
| 3. CONSULTANT SERVICES | | | | **0** | |
| 4. COMPUTER SERVICES | | | | **3,575** | |
| 5. SUBAWARDS | | | | **0** | |
| 6. OTHER | | | | **13,192** | |
| TOTAL OTHER DIRECT COSTS | | | | **17,767** | |
| H. TOTAL DIRECT COSTS (A THROUGH G) | | | | **165,711** | |
| I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **79830 (Rate: 52.0000, Base: 153519)** | | | | | |
| TOTAL INDIRECT COSTS (F&A) | | | | **79,829** | |
| J. TOTAL DIRECT AND INDIRECT COSTS (H + I) | | | | **245,540** | |
| K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.) | | | | **0** | |
| L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K) | | | | $ **245,540** | $ |

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

PI / PD TYPED NAME & SIGNATURE*
**Dan C Marinescu**
DATE

ORG. REP. TYPED NAME & SIGNATURE*
DATE

**FOR NSF USE ONLY**
INDIRECT COST RATE VERIFICATION

| Date Checked | Date Of Rate Sheet | Initials - ORG |
|---|---|---|
| | | |

# SUMMARY PROPOSAL BUDGET YEAR 3

| FOR NSF USE ONLY | | |
|---|---|---|
| PROPOSAL NO. | DURATION (months) Proposed | Granted |
| AWARD NO. | | |

ORGANIZATION: **Purdue Research Foundation**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Dan C Marinescu**

| A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets) | NSF Funded Person-mos. CAL | ACAD | SUMR | Funds Requested By proposer | Funds granted by NSF (if different) |
|---|---|---|---|---|---|
| 1. **Dan C Marinescu - none** | **0.00** | **0.00** | **2.00** | $ **22,453** | $ |
| 2. | | | | | |
| 3. | | | | | |
| 4. | | | | | |
| 5. | | | | | |
| 6. ( **0** ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE) | **0.00** | **0.00** | **0.00** | **0** | |
| 7. ( **1** ) TOTAL SENIOR PERSONNEL (1 - 6) | **0.00** | **0.00** | **2.00** | **22,453** | |
| B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS) | | | | | |
| 1. ( **2** ) POST DOCTORAL ASSOCIATES | **12.00** | **0.00** | **0.00** | **88,259** | |
| 2. ( **0** ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.) | **0.00** | **0.00** | **0.00** | **0** | |
| 3. ( **2** ) GRADUATE STUDENTS | | | | **17,426** | |
| 4. ( **0** ) UNDERGRADUATE STUDENTS | | | | **0** | |
| 5. ( **0** ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY) | | | | **0** | |
| 6. ( **0** ) OTHER | | | | **0** | |
| TOTAL SALARIES AND WAGES (A + B) | | | | **128,138** | |
| C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS) | | | | **27,858** | |
| TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C) | | | | **155,996** | |
| D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) | | | | | |
| TOTAL EQUIPMENT | | | | **0** | |
| E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS) | | | | **4,000** | |
| 2. FOREIGN | | | | **0** | |
| F. PARTICIPANT SUPPORT COSTS | | | | | |
| 1. STIPENDS $ **0** | | | | | |
| 2. TRAVEL **0** | | | | | |
| 3. SUBSISTENCE **0** | | | | | |
| 4. OTHER **0** | | | | | |
| TOTAL NUMBER OF PARTICIPANTS ( **0** ) TOTAL PARTICIPANT COSTS | | | | **0** | |
| G. OTHER DIRECT COSTS | | | | | |
| 1. MATERIALS AND SUPPLIES | | | | **0** | |
| 2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION | | | | **1,000** | |
| 3. CONSULTANT SERVICES | | | | **0** | |
| 4. COMPUTER SERVICES | | | | **3,479** | |
| 5. SUBAWARDS | | | | **0** | |
| 6. OTHER | | | | **9,488** | |
| TOTAL OTHER DIRECT COSTS | | | | **13,967** | |
| H. TOTAL DIRECT COSTS (A THROUGH G) | | | | **173,963** | |
| I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **86047 (Rate: 52.0000, Base: 165475)** | | | | | |
| TOTAL INDIRECT COSTS (F&A) | | | | **86,047** | |
| J. TOTAL DIRECT AND INDIRECT COSTS (H + I) | | | | **260,010** | |
| K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.) | | | | **0** | |
| L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K) | | | | $ **260,010** | $ |

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

| PI / PD TYPED NAME & SIGNATURE* | DATE | FOR NSF USE ONLY |
|---|---|---|
| **Dan C Marinescu** | | INDIRECT COST RATE VERIFICATION |
| ORG. REP. TYPED NAME & SIGNATURE* | DATE | Date Checked / Date Of Rate Sheet / Initials - ORG |

# SUMMARY PROPOSAL BUDGET YEAR 4

| FOR NSF USE ONLY | | |
|---|---|---|
| PROPOSAL NO. | DURATION (months) Proposed | Granted |
| AWARD NO. | | |

ORGANIZATION
**Purdue Research Foundation**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR
**Dan C Marinescu**

| A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets) | NSF Funded Person-mos. CAL | ACAD | SUMR | Funds Requested By proposer | Funds granted by NSF (if different) |
|---|---|---|---|---|---|
| 1. **Dan C Marinescu - none** | **0.00** | **0.00** | **2.00** | $ **23,575** | $ |
| 2. | | | | | |
| 3. | | | | | |
| 4. | | | | | |
| 5. | | | | | |
| 6. ( **0** ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE) | **0.00** | **0.00** | **0.00** | **0** | |
| 7. ( **1** ) TOTAL SENIOR PERSONNEL (1 - 6) | **0.00** | **0.00** | **2.00** | **23,575** | |
| B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS) | | | | | |
| 1. ( **2** ) POST DOCTORAL ASSOCIATES | **12.00** | **0.00** | **0.00** | **91,789** | |
| 2. ( **0** ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.) | **0.00** | **0.00** | **0.00** | **0** | |
| 3. ( **2** ) GRADUATE STUDENTS | | | | **18,297** | |
| 4. ( **0** ) UNDERGRADUATE STUDENTS | | | | **0** | |
| 5. ( **0** ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY) | | | | **0** | |
| 6. ( **0** ) OTHER | | | | **0** | |
| TOTAL SALARIES AND WAGES (A + B) | | | | **133,661** | |
| C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS) | | | | **29,006** | |
| TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C) | | | | **162,667** | |
| D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) | | | | | |
| TOTAL EQUIPMENT | | | | **0** | |
| E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS) | | | | **4,000** | |
| 2. FOREIGN | | | | **0** | |
| F. PARTICIPANT SUPPORT COSTS | | | | | |
| 1. STIPENDS $ **0** | | | | | |
| 2. TRAVEL **0** | | | | | |
| 3. SUBSISTENCE **0** | | | | | |
| 4. OTHER **0** | | | | | |
| TOTAL NUMBER OF PARTICIPANTS ( **0** ) TOTAL PARTICIPANT COSTS | | | | **0** | |
| G. OTHER DIRECT COSTS | | | | | |
| 1. MATERIALS AND SUPPLIES | | | | **0** | |
| 2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION | | | | **1,000** | |
| 3. CONSULTANT SERVICES | | | | **0** | |
| 4. COMPUTER SERVICES | | | | **3,410** | |
| 5. SUBAWARDS | | | | **0** | |
| 6. OTHER | | | | **9,912** | |
| TOTAL OTHER DIRECT COSTS | | | | **14,322** | |
| H. TOTAL DIRECT COSTS (A THROUGH G) | | | | **180,989** | |
| I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **89480 (Rate: 52.0000, Base: 172077)** | | | | | |
| TOTAL INDIRECT COSTS (F&A) | | | | **89,480** | |
| J. TOTAL DIRECT AND INDIRECT COSTS (H + I) | | | | **270,469** | |
| K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.) | | | | **0** | |
| L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K) | | | | $ **270,469** | $ |

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

| PI / PD TYPED NAME & SIGNATURE* | DATE | FOR NSF USE ONLY |
|---|---|---|
| **Dan C Marinescu** | | INDIRECT COST RATE VERIFICATION |
| ORG. REP. TYPED NAME & SIGNATURE* | DATE | Date Checked / Date Of Rate Sheet / Initials - ORG |

# SUMMARY PROPOSAL BUDGET

**Cumulative**

| FOR NSF USE ONLY | | |
|---|---|---|
| PROPOSAL NO. | DURATION (months) Proposed | Granted |
| AWARD NO. | | |

ORGANIZATION: **Purdue Research Foundation**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Dan C Marinescu**

| A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets) | NSF Funded Person-mos. CAL | ACAD | SUMR | Funds Requested By proposer | Funds granted by NSF (if different) |
|---|---|---|---|---|---|
| 1. **Dan C Marinescu - none** | **0.00** | **0.00** | **5.00** | $ **55,895** | $ |
| 2. | | | | | |
| 3. | | | | | |
| 4. | | | | | |
| 5. | | | | | |
| 6. ( ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE) | **0.00** | **0.00** | **0.00** | **0** | |
| 7. ( **1** ) TOTAL SENIOR PERSONNEL (1 - 6) | **0.00** | **0.00** | **5.00** | **55,895** | |
| B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS) | | | | | |
| 1. ( **8** ) POST DOCTORAL ASSOCIATES | **48.00** | **0.00** | **0.00** | **346,512** | |
| 2. ( **0** ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.) | **0.00** | **0.00** | **0.00** | **0** | |
| 3. ( **11** ) GRADUATE STUDENTS | | | | **92,228** | |
| 4. ( **0** ) UNDERGRADUATE STUDENTS | | | | **0** | |
| 5. ( **0** ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY) | | | | **0** | |
| 6. ( **0** ) OTHER | | | | **0** | |
| TOTAL SALARIES AND WAGES (A + B) | | | | **494,635** | |
| C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS) | | | | **102,520** | |
| TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C) | | | | **597,155** | |
| D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) $ **16,000** | | | | | |
| TOTAL EQUIPMENT | | | | **16,000** | |
| E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS) | | | | **16,000** | |
| 2. FOREIGN | | | | **0** | |
| F. PARTICIPANT SUPPORT COSTS | | | | | |
| 1. STIPENDS $ **0** | | | | | |
| 2. TRAVEL **0** | | | | | |
| 3. SUBSISTENCE **0** | | | | | |
| 4. OTHER **0** | | | | | |
| TOTAL NUMBER OF PARTICIPANTS ( **0** ) TOTAL PARTICIPANT COSTS | | | | **0** | |
| G. OTHER DIRECT COSTS | | | | | |
| 1. MATERIALS AND SUPPLIES | | | | **0** | |
| 2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION | | | | **4,000** | |
| 3. CONSULTANT SERVICES | | | | **0** | |
| 4. COMPUTER SERVICES | | | | **14,009** | |
| 5. SUBAWARDS | | | | **0** | |
| 6. OTHER | | | | **48,664** | |
| TOTAL OTHER DIRECT COSTS | | | | **66,673** | |
| H. TOTAL DIRECT COSTS (A THROUGH G) | | | | **695,828** | |
| I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) | | | | | |
| TOTAL INDIRECT COSTS (F&A) | | | | **330,285** | |
| J. TOTAL DIRECT AND INDIRECT COSTS (H + I) | | | | **1,026,113** | |
| K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.) | | | | **0** | |
| L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K) | | | | $ **1,026,113** | $ |

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

PI / PD TYPED NAME & SIGNATURE*: **Dan C Marinescu** DATE

ORG. REP. TYPED NAME & SIGNATURE* DATE

**FOR NSF USE ONLY**

INDIRECT COST RATE VERIFICATION

| Date Checked | Date Of Rate Sheet | Initials - ORG |
|---|---|---|
| | | |

**NSF Form 1030 (10/99) Supersedes all previous editions**

C***SIGNATURES REQUIRED ONLY FOR REVISED BUDGET (GPG III.B)**

## F. BUDGET JUSTIFICATION

We request some support for the PI, and full support for two Postdoctoral Fellow, and a number of Ph.D. students, four in the first year, three in the second and two in subsequent years. There will be no support for the PI in the first year, one summer month in the second year, and full summer support for the last two years of the project. As the only senior member of the group he has a wide range of responsibilities for the management of the Bond project presented in this proposal. We request travel funds to attend various meetings and present the results of our research, for example the Heterogeneous Computing Workshop, Distributed Systems Conferences, Agent Conferences, and so on. We also request $16 K in the first year for equipment. We plan to buy four high end PCs.

The two Post Doctoral Fellows are: Yongchang Ji who got his Ph.D. from the University of Science and Technology of China, in Hefei, in 1998 and Baomin Xu who will get his Ph.D. from the Chinese Academy of Science in Beijing in September 2000. Yongchang will join our group in August 2000 and Baomin in January 2001. The Ph.D. dissertation of Yongchang was in the area of parallel computing and he has published ten papers in this area. The Ph.D. dissertation of Baomin is in the area of distributed systems.

Kyungkoo Jun got his BS from University of Seoul, South Korea in 1995 and an MS degree in CS from Purdue in the Spring of 1998. He passed Qualifying Examinations in May 1998 and Qualifying Examinations in November 1999. Kyungkoo's responsibility is the development of the monitoring framework and of discovery agents. He is also studying the development of a negotiation framework and of broker agents.

Krzysztof Palacz got his BS and MS in Physics and an MS in Computer Science from the University of Krakow in Poland. He passed his Quals in November 1999. Krzysztof's responsibility is the workflow enactment angine and the knowledge management.

Radu Sion got his BS and MS in Computer Sciences from the Polytechnic Institute in Bucharest, Romania and joined the CS Department at Purdue in August 1999. He passed Quals I in May 2000. Radu is working on Web-based agent control.

Tiberiu Steff will join the CS Department in August 2000. He has a BS and an MS in Computer Sciences from the Polytechnic Institute in Cluj, Romania.

# Current and Pending Support

**(See GPG Section II.D.8 for guidance on information to include on this form.)**

The following information should be provided for each investigator and other senior personnel. Failure to provide this information may delay consideration of this proposal.

Investigator: **Dan Marinescu**

Other agencies (including NSF) to which this proposal has been/will be submitted.

---

Support: ☒ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **Parallel and Distributed Computing for Solving Large Structural Biology Problems**

Source of Support: **NSF**

Total Award Amount: $ **2,385,000** Total Award Period Covered: **09/01/95 - 08/31/99**

Location of Project: **W. Lafayette, IN**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad: **0.00** Sumr: **2.00**

---

Support: ☒ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **Enhanced 3D Processing of Spherical and Non-Spherical Virus Structures at High Resolution**

Source of Support: **NSF**

Total Award Amount: $ **675,000** Total Award Period Covered: **09/01/00 - 08/31/03**

Location of Project: **W. Lafayette, IN**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad: **0.00** Sumr: **2.00**

---

Support: ☐ Current ☒ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **Planning and Workflow Management for A Virtual Laboratory for Structural Biology**

Source of Support: **NSF**

Total Award Amount: $ **1,026,114** Total Award Period Covered: **01/01/01 - 12/31/05**

Location of Project: **W. Lafayette, IN**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad: **0.00** Sumr: **0.00**

---

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: $ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

---

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: $ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Summ:

---

*If this project has previously been funded by another agency, please list and furnish information for immediately preceding funding period.

## H. FACILITIES, EQUIPMENT AND OTHER RESOURCES

**Dan Marinescu's lab.**

The equipment in the lab consists of: a Power Challenge SGI system, 2 SGI Indigo 2 Extreme, 6 SGI Indy class machines, 4 SUNs, two Quad Pentium Pro machines, 10 single processor Pentium II systems, and several Apple systems. Four of the systems in the lab have ATM connections, the rest are connected using 100 Mbps Ethernet interfaces. We share a 32 processor Pentium II system with another project and work together with the group of Prof. H.J. Siegel in Electrical and Computer Engineering Department at Purdue and have access to their machines. 1000 sq. feet of space are allocated to our lab. In addition to local computing facilities we have access to computing facilities at Caltech and University of Illinois.

**Computer Sciences Department Computing Facilities**

The Computer Sciences Department has more than 200 SUN, SGI, and IBM workstations and personal computers. An ATM network with several FORE switches and CISCO routers is installed. The Web page at http://www.cs.purdue.edu/facilities/overview.html provides a description of CS facilities.