Distributed and Parallel Databases, 13, 43–72, 2003

IntelliGEN: A Distributed Workflow System for Discovering Protein-Protein Interactions

KRYS KOCHUT
Department of Computer Science, University of Georgia, Athens, GA 30602, USA

JONATHAN ARNOLD
Department of Genetics, University of Georgia, Athens, GA 30602, USA

AMIT SHETH, JOHN MILLER, EILEEN KRAEMER, BUDAK ARPINAR AND JORGE CARDOSO
Department of Computer Science, University of Georgia, Athens, GA 30602, USA

Abstract. A large genomics project involves a significant number of researchers and technicians performing dozens of tasks, either manual (e.g. performing laboratory experiments), computer assisted (e.g. looking for genes in the GENBANK database), or sometimes performed entirely automatically by the computer (e.g. sequence assembly). It has become apparent that managing such projects poses overwhelming problems and may lead to results of lower or even unacceptable quality, or possibly drastically increased project costs. In this paper, we present a design and an initial implementation of a distributed workflow system created to schedule and support activities in a genomics laboratory. The focus of the activities in the laboratory is the discovery of protein-protein interactions of fungi, specifically *Neurospora crassa*. We present our approach of developing, adapting and applying workflow technology in the genomics lab and illustrate it using one distinct part of a larger workflow to discover protein-protein interactions. Novel features of our system include the ability to monitor the quality and timeliness of the results and if necessary, suggesting and incorporating changes to the selected tasks and their scheduling.

Keywords: workflow management, biological process, bioinformatics, protein-protein interaction, laboratory information management

1. Introduction

In the new millennium biology has gone through a paradigm shift with the advent of Genomics, the study of the structure, function, and evolution of whole genomes. The term "genomics" was coined in 1986, and institutional support for this new discipline began with the creation of the National Center for Human Genome Research (NCHGR) in 1989. The initial goal of this discipline was the determination of the entire DNA sequence of a human being (a human genome) and several related model organisms that have played a central role in Genetics, including the bacterium *Escherichia coli*, the yeast *Saccharomyces cerevisiae*, the worm *Canerohabditis elegans*, and the fruit fly *Drosophila melanogaster*. Sequencing the human genome and those of each model system were daunting distributed computing tasks which required collecting, storing, integrating, retrieving, and distributing 3 billion base pairs (bp) of sequence on the human genome alone.

Many aspects of computer science were brought to bear to solve these problems [43, 81]. What genomics promises is that with the availability of the complete genetic blueprints of a variety of living systems, we will be able to unlock how living cells function and evolve. Already genetics has given us the Central Dogma: DNA) Deoxyribonucleic Acid, a long polymer composed of four bases (or letters) makes up the double helix, encodes our genetic blueprint; RNA) genes in our genetic blueprint are transcribed into a related information molecule called RNA to carry the instructions in the blueprint out into the cell; protein) the message is translated into proteins (another information molecule with a 20 letter alphabet), which carry out the work of the cell. The tools of genomics for the first time provide new experiments that describe the complete blueprint (DNA), the cellular levels of all RNAs (RNA profiling), and the cellular levels of all proteins (protein profiling), i.e., not only the information on what a cell is to do (as encoded in the DNA) but what the cell is doing (by RNA and protein profiling).

The paradigm shift in biology is that it is becoming an information science. This means biology is becoming: (1) *data driven* and *informatics-based*, with billions of base pairs of DNA sequence in public databases (http://ncbi.nih.gov), used to collect, store, integrate, retrieve, and distribute the avalanche of genomics data; (2) *high throughput* and *computational*, with new technologies not only to sequence a genome but to describe the full state of the cell as with new technologies like RNA profiling [22]; (3) *hierarchical in its organization* of information along the pathway of the Central Dogma; (4) *inherently mathematical*, in order to have the ability to organize and analyze the data; (5) *systems science-oriented*, with a focus on living systems as complex networks [78]. This paradigm shift is driven by the promise of being able to understand important complex traits at a cellular level. These traits include heart disease and cancer on one hand, and fundamental processes such as metabolism, development, survival, biological clocks, and mating on the other hand, all controlled by many genes at once.

Since 1989, a number of new computational problems have emerged, and computer scientists have played a fundamental role in the birth of this new discipline. Genome projects by their nature are distributed [34]. The overall task of sequencing any genome requires the collaboration of scientists at many sites working on a hierarchically organized project. The scientists take on different well-defined tasks in the project and then transmit the information they produce to other scientists engaged in other tasks to complete the project.

A large genomics project involves a significant number of researchers, technicians, and other support personnel directly involved in the lab activities. All the people perform dozens of tasks, either manual (e.g. performing laboratory experiments), computer assisted (e.g. looking for related genes in the GENBANK database), or sometimes performed entirely automatically by the computer (e.g. genomic sequence assembly). Typically, the tasks must be performed in a specific sequence according to the adopted experimental method. In addition, most of these tasks require vast amounts of support data from a variety of disparate sources, ranging from local flat files, database servers storing experimental results, to a wealth of Web accessible data sources, such as GENBANK and the Protein Database. It has become apparent that managing such projects poses overwhelming problems and may lead to results of lower or even unacceptable quality, or possibly drastically increased project costs.

In this paper, we present a design and an initial implementation of a distributed workflow system created to schedule and support activities in a genomics laboratory focused on what genes do. This project investigates which proteins work together to accomplish tasks in the cell, discovering protein-protein interactions of fungi, specifically *Neurospora crassa*. Our goal is to create a highly flexible workflow system that efficiently coordinates, manages and schedules lab activities, including those involving collection, use and sharing of the necessary experimental data. Our approach of developing, adapting and applying workflow technology is presented in some detail using one distinct part of a larger workflow to discover protein-protein interactions. In addition, novel features of our system include the ability to monitor the quality and timeliness of the results and if necessary, suggesting and incorporating changes to the selected tasks and/or their scheduling.

A primary contribution of this work is in-depth modeling and study of a distinct part of very complex biological process, which allows us to demonstrate the benefit of bioinformatics with the focus on support for process aspects of informatics challenges. In particular, it demonstrates the needs for workflow management to enable faster discovery of biological processes such as protein-protein discovery by enabling effective processing and analysis of laboratory experiment data. Considering just the number of tasks whose execution need to be coordinated shows that such activities are nearly impossible without the automation afforded by workflow management. For the workflow management research, it allows us to show several requirements, including:

- workflows with varied task interconnections and complex data exchanges between tasks
- very large number (tens of thousands) of task executions in a distributed environment involving multiple computers and multiple laboratories,
- processes requiring adaptation to reflect what is learned and addressing new details not addressed earlier, and requiring exception handling to prevent loss of extensive work performed, and
- ability to assess quality of scientific results.

In this paper, we discuss in detail the issues of complexity, quality of service measurement, as well as adaptation. However, our experience in using the protein-protein interaction workflow has not been long enough to discuss comprehensive aspects of exception handling. By using state of the art research in workflow management for a challenging biology problem, this work also demonstrate the current capabilities and future requirements in one instance of collaboration between biologists and computer scientists. While earlier workflow systems have been used to automate laboratory experiments, we believe that IntelliGEN demonstrates a new generation of optimized laboratory workflows that cannot be supported by homegrown (such as script based) or currently available commercial laboratory information systems.

2. Background

2.1. Protein-protein interactions

Genomics is only a beginning in discovering and investigating biological phenomena that drive humans and life. The blueprints are here, but what do they mean? How do we decipher the Rosetta Stone now available on many organisms and make Genomics a hypothesis-driven science? The focus on what cells do will initially follow the path of sequencing simpler systems, such as microbes. Microbes have small genomes with 7 to 49 Mbp of DNA [10], and many of them, like those in the Fungal Kingdom, share properties with their more complex relatives. For example, the filamentous fungus *N. crassa* has a biological clock [54]; however, these simpler microbial systems remain more tractable for analyzing what their cells are doing [21].

One way to describe how living systems function is to think in terms of another metaphor from computer science: a living system is a biological circuit. Each organism is a network of biochemical reactions, as shown in figure 1. In the past the focus of biologists is to carve out one small piece of the larger genomic circuit diagram and focus for 20–30 years on understanding one tiny piece of the biological circuit. A classic example is the Nobel Prize winning work of Jacob and Monod to construct the first biological circuit describing

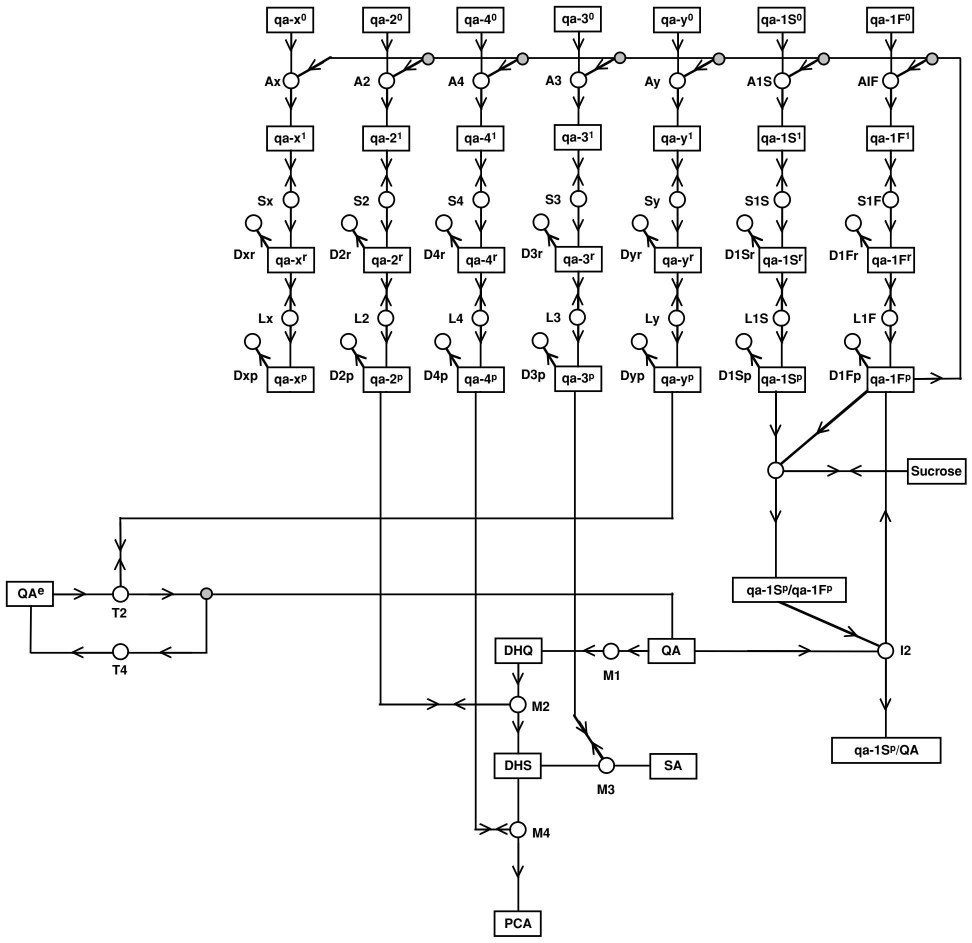


Figure 1. A kinetics model of quinic acid metabolism represented as a biological circuit.

how *E. coli* combusts (i.e., metabolizes) lactose and derives energy from the process. In particular, figure 1 shows one of the early paradigms for eukaryotic gene regulation in *N. crassa* [27], which describes how *N. crassa* metabolizes quinic acid, a compound from which the organism derives energy to live.

The paradigm shift through genomics is to move from one small part of the circuit in figure 1 to the whole circuit. A major goal of genomics is to reconstruct the entire biological circuit for an organism to describe all of its functions. The hope is that by a systems approach we can quickly reconstruct large complex networks and show how these biological circuits can provide predictions about complex traits involving many genes, such as heart disease, cancer, metabolism, biological clocks, development, viability, and mating involving many genes. The reconstruction of such a network requires a diverse array of experimental tasks to be carried out. To make the task concrete we focus now on one small piece of the whole circuit, taking the *qa* cluster as an example. This is appropriate because it was in this model system that the biochemical function of genes was first discovered sixty years ago [9].

The specification of the model in figure 1 begins by writing down the chemical reactions of the known participants in quinic acid (QA) metabolism. The circles on the wiring schematic denote reactions, and the boxes are reactants. Arrows indicate the reactants entering a reaction, and outgoing arrows indicate the products of a reaction. Reactants include the 7 genes in the *qa* gene cluster (*qa-x*, qa-2, *qa-4*, *qa-3*, *qa-y*, *qa-1S*, *qa-1F*) [27] in the boxes at the top of the circuit. These genes can be either in an unbound ('off' state) or a bound state ('on' state) with a protein (i.e., transcriptional activator) produced by the *qa-1F* gene as indicated by a superscript, 0 or 1, respectively. These are in turn transcribed into RNA species (superscripted with an r) carrying the message of the genetic blueprint out into the cell, where in turn the messenger RNAs are translated into protein products (superscripted with a *p*) to carry out the work of the cell. The first four rows of the circuit are simply a restatement of the Central Dogma.

What remains is to specify in the circuit what the proteins are doing. One protein qa-1F^p turns on the circuit, and another protein qa-1S^p turns off the circuit. There are at least 7 protein/DNA interactions between qa-1F^p, and regions near the genes in the *qa* cluster. These protein/DNA interactions determine whether or not a gene is on or off. The bound state leads to activation of all seven genes while the unbound state to inactivation of all seven genes.

Proteins can collaborate as molecular machines to carry out the work in the cell. These collaborations are called protein-protein interactions, and their identification is the whole purpose of the workflow described in this paper. There is one identified protein-protein interaction in the biological circuit between the repressor, qa-1S^p, and the transcriptional activator, qa-1F^p. The repressor protein in some unknown way blocks the activator from working.

The cell must also adapt to its environment and acquire energy to live. QA is the energy source for the cell and is hypothesized to be the cell signal that interacts with the bound complex of qa-1S^p/qa-1F^p to promote induction [27]. When the complex forms, the system is in the off state. The presence of QA switches the system from the off to on state by favoring the unbound state of the transcriptional activator.

In the lower half of the circuit a total of 4 out of the 7 protein products participate on a known biochemical pathway ultimately converting QA into products feeding into the energy producing Krebs Cycle. There are at least two cellular states for QA, extracellular (denoted with an e) or intracellular QA. The cell only goes to work on QA when it is imported into the cell. One of the genes, *qa-y*, produces a permease, qa-y^p, which is thought to be involved in the transport of QA into the cell.

For most reactions, mass balance leads to a specification of a system of differential equations to describe this reaction network or biological circuit [12]. The boxes in the middle of this diagram with the RNA and protein species are the observables. The boxes at the bottom are the reactants in the underlying biochemical pathway. The model is a first approximation of what needs to be considered to predict induction of the *qa* cluster and its products. This model is a highly simplified version of what the cell is actually doing with the QA. We have not put the molecular machine that transcribes DNA into messenger RNA. Neither have we put in the molecular machine that translates messenger RNA into protein. As a first approximation, transcriptional and translational machinery are treated as an infinite resource, available as needed. The aim is to introduce only enough of the "wiring schematic" of the organism into the model to make the model predictive about how the system responds when the system is perturbed genetically, environmentally, or chemically.

The model is then fitted to the observed RNA and protein profiles (the boxes in the third and fourth rows of the circuit) and evaluated for fit in the presence of varied perturbations. The system is perturbed genetically when the *qa-2* gene is knocked out, and the system observed. The system is perturbed chemically when an inhibitor is added to inhibit the qa-2^p gene product, and the system is observed. The system is perturbed environmentally when sucrose is added or removed as the preferred carbon source, and the system is observed. In each perturbation the circuit is simulated with a reaction network simulator, which leads to predictions of the messenger RNA and protein profiles over time. The predicted RNA and protein profiles either match or do not match the observed profiles. In all likelihood, it will be necessary to add additional components to the wiring schematic in figure 1 even in this well-understood paradigm of eukaryotic gene regulation [27].

Imagine extending this whole process to the cell. One starting point may be something like a cell with only 256 genes [42] in a very small microbial genome. Can we build the circuit and show that the circuit describes how the cell functions? This is hypothesis-driven genomics. To carry out this program on a model microbial system requires the completion of different tasks with the genetic blueprint in hand, including:

1. genetic, chemical, or environmental perturbation of the cell,
2. RNA and protein profiling to describing the state of the cell after perturbation,
3. building protein/protein and protein/DNA interaction maps to build the links in the biological circuit [82],
4. fitting the kinetics models to observed messenger RNA and protein profiles,
5. evaluating the fit of the model,
6. searching for an improved model,
7. storing the intermediate circuit model for later query; and
8. repeating the process.

We model this process as an automated workflow. In the next several sections we will describe one distinct part of this larger workflow, identifying the protein-protein links in the circuits. It is clear that there are many subtasks that will be carried out by different groups of researchers. Then there is the challenge of integrating the information to carry out the fitting process, not to mention the computational task of fitting a large reaction network.

As with sequencing the human genome, the process of computing life, i.e. identifying the biological circuit, involves many new computational problems. One of these is constructing software that allows the design, construction, execution, management, and adaptation of workflows to carry out circuit identification or some subset of the tasks needed for circuit identification. The experiments require typically over 100,000 task executions. The tasks are complex, and as the project unfolds over the course of several years new technologies become available, or discoveries are made that require the workflow to be adapted. Data routinely come from multiple sources in multiple storage media. There is also the challenge of database integration and efficient storage and integration of data with complex structure. The problem of constructing a map of all the protein-protein interactions requires new algorithms similar to the ones in physical mapping [82]. Once the data are integrated, novel algorithms are needed for simulating the large reaction networks, which are at the heart of the data integration step.

2.2. Process management and workflow systems

A workflow is an activity involving the coordinated execution of multiple tasks performed by different processing entities [53]. These tasks could be manual, or automated, either created specifically for the purpose of the workflow application being developed, or possibly already existing as *legacy* programs. A workflow process is an automated organizational process involving both human (manual) and automated tasks.

Workflow management is the automated coordination, control and communication of work as is required to satisfy workflow processes. A Workflow Management System (WfMS) is a set of tools providing support for the necessary services of workflow creation (which includes process definition), workflow enactment, and administration and monitoring of workflow processes [39]. The developer of a workflow application relies on tools for the specification of the workflow process and the data it manipulates. The specification tools cooperate closely with the workflow repository service, which stores workflow definitions. The workflow process is based on a formalized workflow model that is used to capture data and control-flow between workflow tasks.

The workflow enactment service (including a workflow manager and the workflow runtime system) consists of execution-time components that provide the execution environment for the workflow process. A workflow runtime system is responsible for enforcing inter-task dependencies, task scheduling, workflow data management, and for ensuring a reliable execution environment. Administrative and monitoring tools are used for management of user and work group roles, defining policies (e.g., security, authentication), audit management, process monitoring, tracking, and reporting of data generated during workflow enactment.

Workflow technology has matured to some extent, and current products are able to support a range of applications (for technology and state of the art overview, see [2, 23, 28]. Nevertheless, many additional limitations remain, especially in supporting more demanding applications, more dynamic environments and for better support for human involvement in organizational activities and better support for Quality of Service (QoS) management [17, 73]. In this paper, we focus on problems involved in supporting a large genomics project, in which a number of additional demands are placed on the workflow management system. These demands include high adaptability to varying experimental conditions in the lab, automatic quality assessment of the experimental results, as well as assisting the workflow administrator and researchers in introducing changes in the workflow due to inadequate quality or timeliness of the results. Research issues based on these requirements have been investigated as part of the METEOR project and workflow management system developed at the LSDIS lab of the Computer Science Department at the University of Georgia [58], which we use in this effort.

2.3. Fungal Genome Database

The Fungal Genome Database (FGDB) [50, 69] is a 10-year development effort that supports storage, retrieval, and distribution of all of our data over the Web [40] for physical mapping, genetic mapping, sequencing, and now protein-protein interaction mapping experiments. An important scientific contribution of FGDB is its support of ordered sequences of genomic objects in order to meet the efficient computation requirements involving genome data. FGDB is implemented in the object-relational database system, Oracle 8*i* Enterprise Edition, on a SUN Enterprise 250 server. The outline of the database schema is presented in figure 2 in the form of a UML (Unified Modeling Language) class diagram [71].

We have also developed Java-based tools to visualize the data in FGDB, which can be downloaded from our web site [33, 79, 87, 88]. FGDB supports physical mapping of *Aspergillus nidulans, N. crassa, Aspergillus flavus, Nectria haematococca*, and *Pneumocystis carinii* and sequencing projects for *N. crassa* and *P. carinii*.

2.4. Workflow management system METEOR

METEOR is a comprehensive Workflow Management System. METEOR's architecture includes a collection of four services: Workflow Builder, Workflow Repository, Workflow Enactment, and Workflow Manager. Workflow Enactment includes two services: ORB-Work and WebWork. Both ORBWork and WebWork use fully distributed implementations. WebWork [61], an entirely Web-based enactment service, is a comparatively light-weight implementation that is well-suited for a variety of enterprise workflow process applications that involve limited data exchange and do not need to be dynamically changed. ORBWork [51] (used in this project) is better suited for more demanding, mission-critical enterprise applications requiring high scalability, robustness and dynamic modifications. The overall architecture of the system is shown in figure 3.

2.4.1. Workflow builder service. This service consists of a number of components that are used to design graphically and specify a workflow, in some cases leaving no extra work after a designed workflow is converted to a workflow application by the runtime code generator. Its three main components are used to specify the entire map of the workflow, data objects manipulated by the workflow, as well as the details of task invocation, respectively. The task

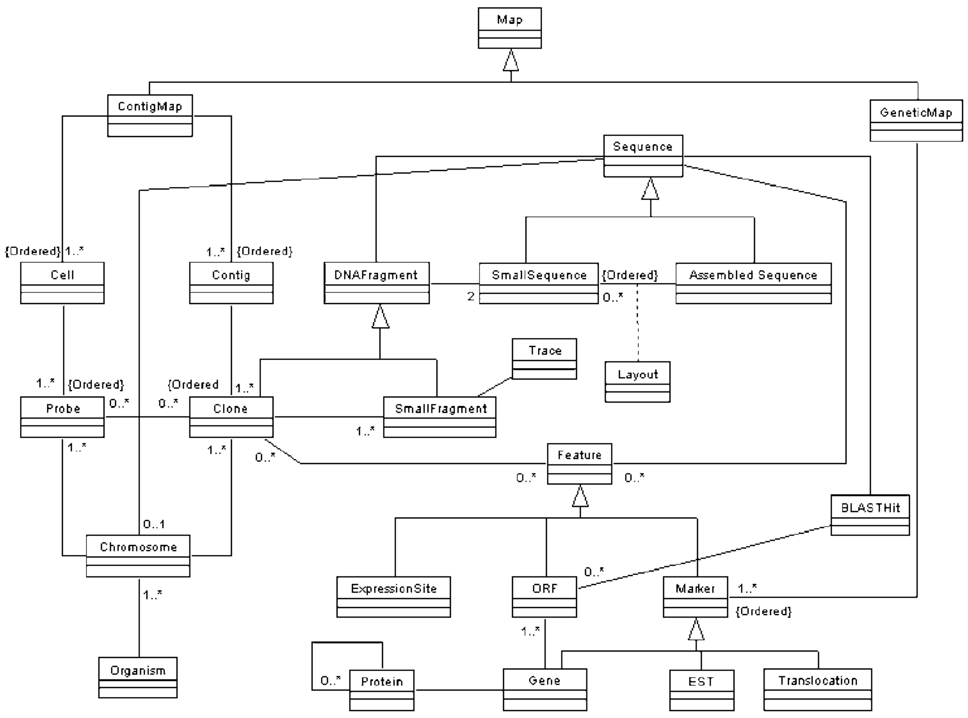


Figure 2. Database schema (more detailed schema at [40]).

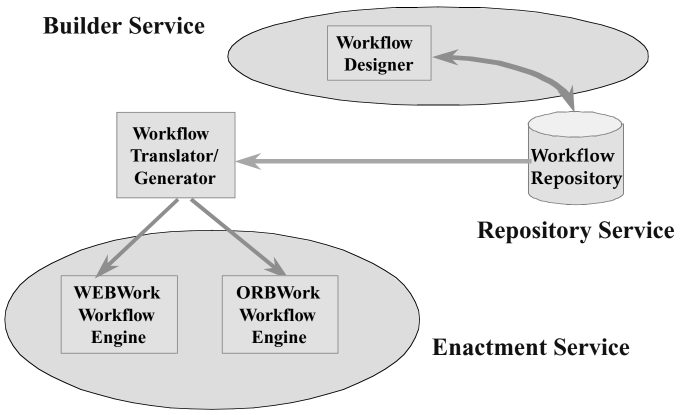


Figure 3. METEOR architecture.

design component provides interfaces to external task development tools (e.g., Microsoft's FrontPage to design the interface of a user task, or a rapid application development tool). This service supports modeling of complex workflows consisting of varied human and automated tasks in a conceptual manner using easy-to-use tools. In particular, the designer of the workflow is shielded from the underlying details of the infrastructure or the runtime environment. At the same time, very few restrictions regarding the specification of the workflow are placed on the designer.

The workflow specification created using this service includes all the predecessor-successor dependencies between the tasks as well as the data objects that are passed among the different tasks. It also includes definitions of the data objects, and the details of task invocation. The specification may be formatted to be compliant with the Workflow Process Definition Language (WPDL) of the Workflow Management Coalition [39] and its subsequently defined XML syntax. This service assumes no particular implementation of the workflow enactment service (runtime system). Its independence from the runtime supports separating the workflow definition from the enactment service on which it will ultimately be installed and used. Workflow process definitions are stored in the workflow repository.

2.4.2. Workflow repository service. The METEOR Repository Service is responsible for maintaining information about workflow definitions and associated workflow applications. The graphical tools in the workflow builder service communicate with the repository service and retrieve, update, and store workflow definitions. The tools are capable of browsing the contents of the repository and incorporating fragments (either sub-workflows or individual tasks) of already existing workflow definitions into the one currently being created. The repository service is also available to the enactment service (see below) and provides the necessary information about a workflow application to be invoked.

The first version of the repository service was based on the Interface I API, as specified by WfMC [39]. Subsequently, we have built the second version of the workflow repository [8], in which workflows are stored as XML-documents to facilitate their Web-interchange on a distributed system managed by METEOR. The researcher (or a service of the METEOR system itself) can query the workflow repository in order to introduce dynamic changes needed for workflow adaptation, as described later.

2.4.3. ORBWork enactment system. The task of the enactment service is to provide an execution environment for processing workflow instances. Both ORBWork and WebWork have suitable code generators that can be used to build workflow applications from the workflow specifications generated by the design service or those stored in the repository. In the case of ORBWork, the code generator outputs specifications for task schedulers (see below), including task routing information, task invocation details, data object access information, user interface templates, and other necessary data. The code generator also outputs the code necessary to maintain and manipulate data objects created by the data designer. The task invocation details are used to create the corresponding "wrapper" code for incorporating legacy applications with relative ease. The management service supports monitoring and administering workflow instances as well as configuration and installation of the enactment services.

2.5. *Adaptability and dynamic workflows*

Recently, there has been an increasing interest in developing WfMSs capable of supporting adaptive and dynamic workflows. Such systems must be uniquely sensitive to a rapidly changing process execution triggered by collaborative decision points, context-sensitive information updates, and other external events. The majority of current work addresses relevant issues at modeling and language levels [24, 36, 45, 53, 57, 70] while the relevant issues involving organizational changes appear in [24, 38]. A particularly different approach to supporting adaptive workflow (capable of reacting to the changes in local rules and other conditions) has been developed using the notion of migrating workflows) [19]. Related issues of integrating workflow or coordination technologies and collaborative technologies are investigated in [31, 72].

ORBWork utilizes a fully distributed scheduler in that the scheduling responsibilities are shared among a number of participating *task schedulers*, according to the designed workflow map [51]. Each task scheduler receives the scheduling specifications at startup from the Workflow Repository (currently, the repository service sends the specifications via the HTTP protocol). Each set of task specifications includes the input dependency (input transitions), output transitions with associated conditions, and data objects sent into and out of the task. In the case of a human task (performed directly by an end-user), the specifications include an HTML template of the end-user interface page(s). In the case of a non-transactional automatic task (typically performed by a computer program), the specifications also include a task description and the details of its invocation. Finally, in the case of a transactional task, the specification includes the details of accessing the desired database and the database query.

When a task is ready to execute, a task scheduler activates an associated task manager. The task manager oversees the execution of the task itself. Figure 4 presents a view of the ORBWork's distributed scheduler. Note that scheduling components and the associated tasks and task managers are distributed among four different hosts. The assignment of these components to hosts can be modified at runtime.

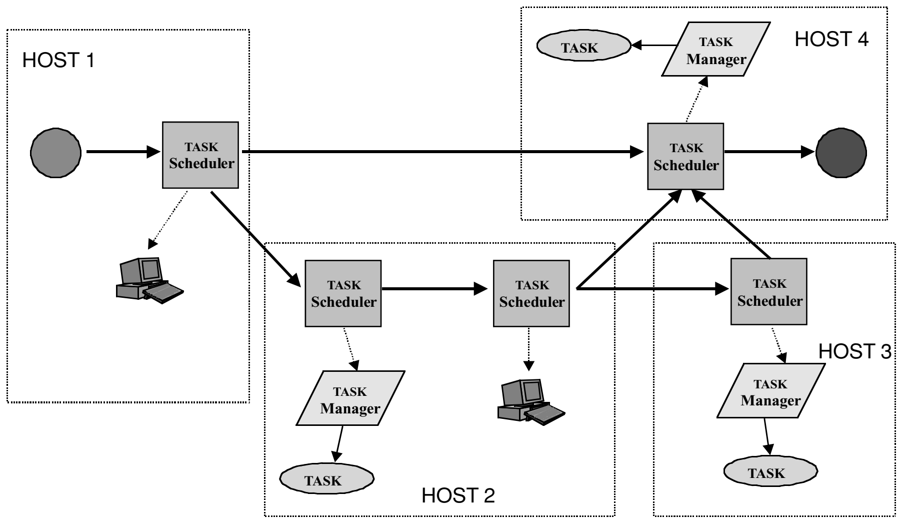


Figure 4. ORBWorks distributed scheduler.

The partitioning of various components (scheduler's layout), including task schedulers, task managers and tasks, among the participating hosts is flexible. An ORBWork administrator may move any of the components from one host to another. In the fully distributed layout, it is possible to place all of the workflow components on different hosts.

Each task scheduler provides a well-constrained subset of the HTTP protocol, and thus implements a lightweight, local Web server. This enables an ORBWork administrator to interact directly with a selected task scheduler and modify its scheduling specifications from a common Web browser. It also enables the end-user to access workflow instances residing on the task's worklist. This set up naturally supports a mobile user.

The ORBWork scheduler and its supporting components have been designed in such a way that the enactment service can be used to support a variety of dynamic changes both to the workflow schema and to the individual workflow instances. The workflow administrator can easily modify the workflow schema at runtime by acquiring new information from the workflow repository, or even by modifying the specification by direct interaction with the scheduler.

We divide the dynamic changes in ORBWork into two categories: *primitive changes and composite changes.* A *primitive change* is composed of "atomic" changes that can only be applied to a process definition totally or not applied at all (e.g., adding a synchronous transition between two tasks). A *composite change* is composed of a sequence of primitive changes that describe a complicated process definition change (e.g., adding a task between two existing tasks can be achieved by applying a sequence of primitive changes as we will see in the following sections). Primitive changes can be further divided into *immediate changes* and *incremental changes*. *Immediate changes* are changes that can be introduced into workflow run-time in one step without losing the correctness and consistency of the workflow. In the context of ORBWork run-time, one step means reloading the necessary process definition files. On the other hand, there are situations when we cannot apply the changes to a particular task in "one shot". Consider that we want to change the input/output dependencies of a task, where several workflow instances are pending on this task (waiting for necessary transitions from the predecessor tasks in order to invoke the task). If we just update the task specifications without taking care of all these already existing workflow instances, they may work incorrectly. *Incremental changes* address that problem. Such changes are introduced into the workflow enactment system step by step and guarantee the correctness and consistency of the whole workflow system. In practice, most of the primitive changes in a workflow system are incremental.

Another very important issue of implementing a dynamic workflow system is how should different *versions* of a workflow/task schema the workflow enactment system support. We say that a particular task is in a stable state if all input/output dependencies, input/output parameters of the workflow instances residing on that task scheduler, are the same. Consider the following scenario: A workflow system is normally running and with several instances working simultaneously. The workflow administrator decides to do some changes to the input dependencies of a task and several instances are under the control of this task's scheduler. From the earlier discussion, we know that some instances should still use the old input dependency schema while new instances should use the new version of the input dependencies. At some time, the task scheduler may be scheduling two workflow instances with different input dependencies. In such a case, the task is unstable. Moreover, if we try to change the input dependencies of that unstable task, the task scheduler will finally have three different versions of input dependencies. If the administrator keeps making changes, the task scheduler may have four, five, six or more input dependency versions.In our current implementation, we only allow two versions of a process definition to exist for the workflow instances residing on a particular task.

An additional issue worth mentioning here is how to suspend the task scheduler. When dynamic changes are introduced to a particular task, we will force the ORBWork runtime to *suspend* that task scheduler. In our implementation, we divide the suspend operation into three different types: *suspend input transition; suspend output transition; suspend both input/output transitions*. After applying the "suspend input transition" operation, no workflow instance is allowed to "flow" to this task by making a transition call on this task's scheduler. Similarly, the "suspend output transition" operation keeps any existing workflow instance on that task from making a transition call to a successor task's scheduler. The third suspend operation is the combination of the previous two.

A detailed description of possible changes and how they are implemented is described in [18]. The types of dynamic modifications currently offered in ORBWork are presented in Table 1. However, sometimes a predefined schedule of tasks may need to be altered for just a single workflow instance, without introducing permanent changes to the workflow schema. The ORBWork process manager allows the per-instance changes of similar types as described above, but only those associated with a single instance, rather than with the whole workflow schema. The changes cease to exist, once the instance completes. Theoretical aspects of introducing dynamic changes to workflow systems are examined in [1].

2.5.1. Support for scalability and fault tolerance. The fully distributed architecture of ORBWork yields significant benefits in the area of scalability. As mentioned, all of the workflow components of a designed and deployed workflow (this includes individual task schedulers, task managers, and task programs) may be distributed to different hosts. However, in practice it may be sufficient to deploy groups of less frequently used task scheduler/manager/programs to the same host. At the same time, heavily utilized tasks may be spread out across a number of available workflow hosts, allowing for better load sharing.

The features of ORBWork designed to handle dynamic workflows are also very useful in supporting scalability. As load increases, an ORBWork administrator may elect to move a portion of the currently running workflow to a host (or hosts) that become available for use in the workflow. The migration can be performed at the time the deployed workflow is running. Simply, the workflow administrator may suspend and shutdown a given task scheduler and transfer it to a new host. Because of the way task schedulers locate their successors, the predecessors of the moved task scheduler will not notice the changed location of the task. If the associated task must be executed on a specific host (for example it is a legacy application), the associated task manager may be left in place, while only the scheduler is transferred.

In the case that a group of task schedulers is deployed to the same host, the ORBWork administrator has the option to combine them into a single "master" scheduler. Such a master

Table 1. Types of dynamic modifications available in ORBWork.

Change type	Change type	After the change
AND to OR Join	Incremental	A single predecessor tasks needs to be completed in order to execute a given task
OR to AND Join	Immediate	All of the predecessor tasks need to be completed in order to execute a given task
AND to OR Split	Immediate	A single successor task will be activated after a given task completes
OR to AND Split	Immediate	All successor tasks will be activated after a given task completes
Add AND Transition	Incremental	One more task will be activated after a given task completes
Add OR Transition	Immediate	One more task may be activated after a given task completes
Delete Transition	Incremental	A given transition will not be attempted (either AND or OR)
Add Object Transfer	Incremental	One more data object will be transferred along a given transition
Delete Object Transfer	Incremental	A data object will not be transferred along a given transition
Parameter Mapping Change	Incremental	An incoming data object will be assigned to a different parameter of a given task
Parameter Type Change	Incremental	A given task will accept a new data object type for a given parameter
Task Type Change	Incremental	A different task type (e.g. *automatic* instead of *human*) will be invoked
Task Invocation Change	Composite	A different task will be invoked (but within the same task type)
Insertion of a Task	Composite	A new task will be performed, if enabled
Deletion of a Task	Composite	A given task will not be performed

scheduler controls a number of individual task schedulers that share the same heavyweight process. This allows the administrator to control the utilization of the participating host even further, while having many individual operating system-level processes (task schedulers) could potentially burden the host system.

The distributed design of ORBWork offers no single point of failure for an ongoing workflow instance. Since the individual task schedulers cooperate in the scheduling of workflow instances, a failure of a single scheduler does not bring the whole system down, and other existing workflow instances may continue execution.

The error handling and recovery framework for ORBWork [84] has also been defined in a scalable manner. All errors are organized into error class hierarchies, partitioning the recovery mechanism across local hosts, encapsulating and handling errors and failures as close to the point of origination as possible, and minimizing the dependence on low-level operating system-specific functionality of the local computer systems. Complementary work on exception handling, especially on finding alternatives to deal with exceptions, is described in [55].

3. Discovering protein-protein interactions

With the completion of the sequencing of the human genome and that of other model systems a major new direction has been the characterization of the proteome, the collection of all proteins in the cell, to figure out what cells are doing besides data storage. The genetic blueprint is here. The genome is known. What functions does the genome encode and program through the Central Dogma?

One new direction is to identify all of the proteins produced by the genetic blueprint. In this way we obtain a task list for the organism. This effort has led in a number of directions because in many ways protein structure is much richer than that of DNA. One direction is simply to isolate and characterize all the proteins in the cell. Isolating proteins allows biochemists to examine their function.

From here several directions can be chosen. One direction has been identifying the structures of all proteins in the cell [77]. High-throughput methods for obtaining molecular structures on all proteins are being developed. These molecular structures provide valuable insights into how proteins carry out their tasks. Proteins, unlike DNA, have a vast repertoire of structures to carry out the diversity of functions.

Once the proteins are identified and characterized, a second interest is how they assemble into the molecular machines that carry out the work in the cell. Some of these larger cooperative structures in cell have names like the transcriptosome, splicesome, proteasome, ribosome, cytoskeleton, mitochondrion, circadian clock, spindle, and MAP kinase cascades to carry out basic processes in the cell like transcription, RNA splicing, translation, energy metabolism, cell division, and signaling [82].

Identifying all of the protein-protein interactions is fundamental to searching for connections relevant to a particular process, such as the link qa-1S^p/qa-1F^p in the biological circuit of figure 1. Knowing which proteins work together is part of specifying the biological circuit describing a particular biological process. The collection of protein-protein interactions can be visualized as a map, in which proteins are the nodes and the edges are the interactions (figure 5). A protein-protein interaction network or map then represents a search grid on which biological circuits are constructed. The map tells the researcher what connections he or she may need to consider in the circuit.

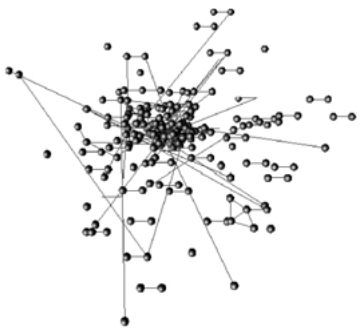


Figure 5. Protein-protein interaction map of *S. cerevisiae* from [44] visualized as a “protein mobile”.

We refer to this Calder-like visualization of a protein-protein interaction map as a "protein mobile". The goal of this paper is to describe a distributed automated workflow to generate this protein mobile accessible over the Web [26, 52]. The example shown is part of the protein-protein interaction map for the yeast *S. cerevisiae* [44]. Eight composite steps comprise the workflow to generate such a map.

Step 1(GetGenes). In eukaryotic systems like humans a major complication is identifying all of the genes that produce the proteins. The complication is that genes in eukaryotes contain regions of DNA called introns, which are not transcribed. The introns are cut out of primary transcript to form mature transcript. A geneticist can isolate all the RNAs in a cell and reverse the process of transcription with the enzyme reverse transcriptase to make complementary DNAs or cDNAs that identify correctly what DNA sequence is ultimately used to make a protein. These cDNAs can be used to create a library of clones called a cDNA library. The first step in the process of identifying all protein-protein interactions is to make a large cDNA library that contains most of the genes in the organism with the introns conveniently spliced out. Ultimately, this cDNA library can be used to make the proteins needed to test for interactions among them.

The main limitation of this strategy to get to the DNA sequence encoding a protein is that cDNA libraries typically do not have all of the genes. Alternative strategies are resorted to. One of these is computational. A large clone is sequenced, and algorithms for gene identification are utilized based on the grammar of DNA to identify genes [52]. Then the genes are extracted directly from the clone by a technique known as polymerase chain reaction, a way to amplify a specific region of DNA from a DNA source like a clone.

Step 2 (GenExpLib sub-workflow). The next step is to build an interaction detector. A standard way to detect interactions is the yeast *S. cerevisiae* 2-hybrid system. The goal is to use proteins in the cell to reconstitute a transcriptional activator like *GAL4* in *S. cerevisiae* and to hook up the transcription factor to a collection of reporter genes which come on only when the protein-protein interaction is present. The *GAL4* gene has two parts, an activation domain (AD) and a binding domain (BD). The AD-domain interacts with another protein to turn on transcription. The binding domain binds to the DNA to activate transcription. The reporter genes are put downstream of the AD-domain to report transcription.

To test for an interaction, one cDNA is fused to the AD-domain. Another cDNA is fused to the BD-domain. If the two cDNAs ultimately produce proteins that interact, then the activation domain (AD) will be brought together with the binding domain (BD) to reconstitute the GAL4 protein, and transcription will be initiated. The library of cDNAs fused to the AD-domains is referred to as the library of prey clones. They are the key that enters the lock. The library of cDNAs fused to the BD-domains is referred to as the library of bait clones. They are the lock waiting for the key. When the bait and prey come together through the protein-protein interaction, the *GAL4* protein is reconstituted, and transcription initiates. In summary, Step 2 is to build the bait and prey libraries.

Step 3 (GenExpLib sub-workflow). *Neurospora crassa* has $\sim$11,000 genes, and it is not possible to screen one by one for the 121,000,000 possible interactions. Instead we use the fact that most proteins do not interact with each other. Instead of screening each pair of potential interactors one at a time, we create pools of bait or prey. There are three pooling strategies that have been used to date.

96 prey encounter 96 bait. In this strategy pools of 96 bait and 96 prey clones are created separately, and the pools are ultimately tested against each other [44]. Each such experiment tests for one or more interactions in a pool of ∼10,000 interactions. In this way [44] screened about 10% of the 36,000,000 possible interactions in the yeast *S. cerevisiae.*

1 prey encounters an array of all baits. In this strategy all bait clones are robotically arrayed on solid media where the individual proteins are expressed [80]. In the case of *N. crassa*, this would mean arraying up to 11,000 different genes on solid media and introducing one prey at each point on the array to test for the interaction. This approach is more easily automated than the first strategy.

All prey encounter 96 baits. In this strategy a mixture of all prey clones in the prey library is created and then tested against a plate of of 96 baits [80]. The entire prey library is allowed to interact with each of the 96 baits individually. This protocol constitutes a high-throughput screen. The plates of 96 baits can be processed robotically. The pool of prey is large. This allowed the creation of the first protein-protein interaction map for the yeast *S. cerevisiae* [80]. The limitation is that there are many more false positives in this screen than strategies 1 or 2. This strategy provides a rough sketch of the map, while the first or second strategy provide detailed sketching.

In that there is a mixture of strategies available, the workflow needs to be adaptive. First, the entire portrait of the protein-protein interaction map needs to be obtained, and then the details need to be sketched in. As interesting connected subsets in the map are uncovered, likely to correspond to interesting molecular machines, a switch needs to be made to a more detailed sketching process. Also, the workflow needs to be adaptive in the sense that new technologies will come on line to detect protein-protein interactions more effectively, and these new technologies need to be introduced into the workflow. Finally, each detector is characterized in part by its false positive rate and false negative rate in detecting interactions. As researchers gain more experience in building these maps, there will be an evolution in quality standards that will also mandate alterations in the workflow.

Step 3 of the workflow is to create the bait and prey pools of cDNAs, which are ultimately used to test for a protein-protein interaction.

Step 4 (IDRemGen and InterMating). In Step 4, the bait and prey pools of clones are brought together to detect the interaction. The mechanism for bringing them together is called an interaction mating [41]. A female strain (α) of the yeast *S. cerevisiae* is transformed with the pool of bait clones; a male strain (a) of the yeast *S. cerevisiae* is transformed with the pool of prey clones. Transformation is the process of introducing foreign DNA into a host; the strains of *S. cerevisiae* are then capable of expressing the proteins of interest. The female and male strains are mated to bring the bait and prey pools together. In strategy 2 this means simply pinning robotically each bait strain on the solid media with the prey strain. Those grid points with the reporter genes on can be visually scored on the array. Step 4 is the interaction mating bringing bait and prey together. The resulting images of the arrays or the positives on plates can be digitally captured.

A number of controls are introduced at this point to confirm the interaction. Three reporter genes exist downstream of the BD ('bait') gene, and each of the reporters give a vote on whether or not the interaction is real. A separate experiment highlights the vote of each reporter gene. In strategy 2, for example, three arrays need to be generated, one for each reporter gene to score visually whether or not a particular reporter gene votes yes to the interaction.

It is possible to alter the threshold of the detector for one reporter gene simply by adding an inhibitor called 3AT to poison the protein product of one reporter gene. The presence of the inhibitor means that the detected protein-protein interaction must be stronger to counteract the effects of the inhibitor. In strategy 2, each threshold selected as indexed by the 3AT concentration used in the solid media operates the detector at a different threshold.

Lastly, different protein pairs may or may not interact in the yeast *S. cerevisiae*. It is possible to repeat the whole experiment in a different host like *E. coli* in order to reduce false negatives in the interaction detector.

As a consequence, the workflow is inherently adaptive depending on the structure of the protein-protein interaction map, the interesting features in the uncovered map, and what regions of the table of all possible interactions are missing.

Step 5 (RobotPickColonies). In Step 5, we need to identify what genes are positive in the pools. In strategy 1, we do not know which of the 96 prey reacted with which of the 96 baits. The positives are robotically picked. The DNA of the positives is extracted and sequenced. By comparing the resulting sequences of the bait and prey clones, we can positively identify the partners that are interacting. These sequences are sometimes referred to as Interaction Sequence Tags (ISTs), and they allow screening for the protein-protein interactions based on the availability of the genomic sequence of the organism of interest. Step 5 is the identification of the interactors by sequencing.

Step 6 (FGDB). In Step 6, the interactors identified through their ISTs are loaded into the Fungal Genome Database FGDB [50]. The FGDB database is web-accessible at http://gene.genetics.uga.edu to the scientific community to make use of the information. Step 6 involves storing, retrieving, releasing, and sharing the ISTS over the Web from FGDB.

Step 7 (Layout). Once the ISTs are available, the data are ready to be assembled into a protein-protein interaction map. The strongly connected components in the graph will allow the identification of putative protein complexes. Several algorithms for laying out these maps have been coded in Java and tested [88]. While the ISTs can be stored as tables of all the directional pairwise interactions in a relational database, the only way to begin to make sense of this information is graphically. A critical intermediate step in visualization is assembling the graph describing the protein-protein interactions and highlighting its features on the graph. The nodes of the graph are proteins, and the directed edges are the interactions. The graph captures the biology and is ultimately rendered in various forms like the protein mobile in figure 5, which is of great interest to biologists. There are many difficult algorithmic problems with identifying this graph. Interacting with the graph is believed to be key to locating biologically relevant clusters. Step 7 is assembling the protein-protein interaction map as represented in a graph.

Step 8 (J3DV). The last step in the workflow is visualizing the protein-protein interaction map over the Web. To this end a Java-based server was created to provide the interface between FGDB and Java objects for map rendering [26]. Second, a Java 3D client software was created to visualize the map [79, 88]. The map is rendered over the Web and provides a view of the data allowing adaptation of the workflow and interpretation of the protein-protein interaction mapping data. The last step is visualizing the protein-protein interaction map before returning to Step 1 to continue map construction.

Each of the composite tasks above contain about 10 individual tasks, and so in Strategy 3, for example, it would be necessary to execute an instance through the workflow about 500 times. Each instance has at least one control. We are talking about managing the execution of ~75,000 tasks with an automated workflow. The tasks themselves are distributed between several experimental locations and over several computers and robots. For example, the libraries are generated in one laboratory, and the robots are located in a different laboratory. The FGDB is located on one server, and the map assembly routine is located on a different server. Image capture involves other workstations.

4. IntelliGEN: Workflow for protein-protein interaction discovery

IntelliGEN is a comprehensive system for genomic data and process management. It implements the overall workflow, as described above. It reuses in part two of our earlier works: (b) GeneFlow [34] build as part of a laboratory information system for managing distributed high throughput sequencing, which supports steps 6 through 8 of the overall workflow, and (b) graphical tools to visualize the mapping and sequencing data [33]. The graphical database tools also support XML messaging to exchange genomic information with other databases and applications [87]. While earlier workflow systems have been used to automate laboratory experiments [16, 29], we believe that current advances in adaptive workflow technologies can improve dramatically the quality of experiments by optimizing laboratory workflows.

In the near term, the core objective of the proposed system is running protein-protein interaction mapping workflows. However, we plan to use the system in other types of genomic workflows to automate identification of a biological circuit. The rest of this section contains a brief discussion of the specific capabilities of IntelliGEN. The architecture of IntelliGEN is shown in figure 6.

IntelliGEN's workflow network is presented in figure 7 (GeneFlow is a subworkflow of this workslow; its tasks are not shown), and a subset of this workflow is used to process the ISTs (Interaction Sequence Tags). These workflow networks are presented here as screen shots of METEOR's Builder service. The top-level protein interaction workflow includes some high-level network tasks (which include further tasks, acting as subworkflows), such as GetGenes, GenExpLib, IdRemGen, InterMating, and RobotPickColonies. These high-level steps correspond to getting genes from a cDNA library or cosmid library, generating expression libraries by recombinational cloning, eliminating spontaneously activating genes from the mapping experiments, performing interaction matings and finally robotically screening (picking) positive interactions, respectively. These tasks are further

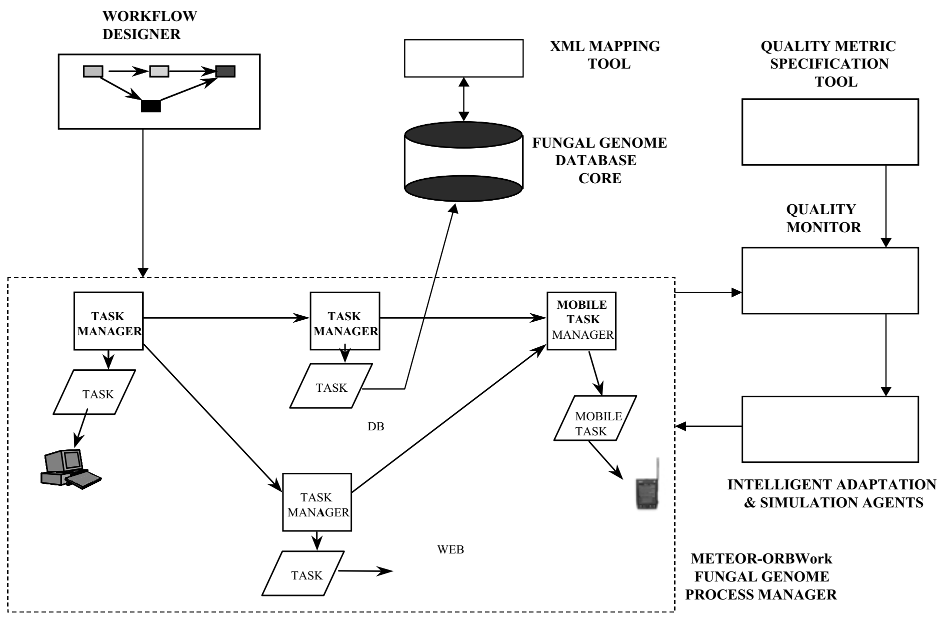
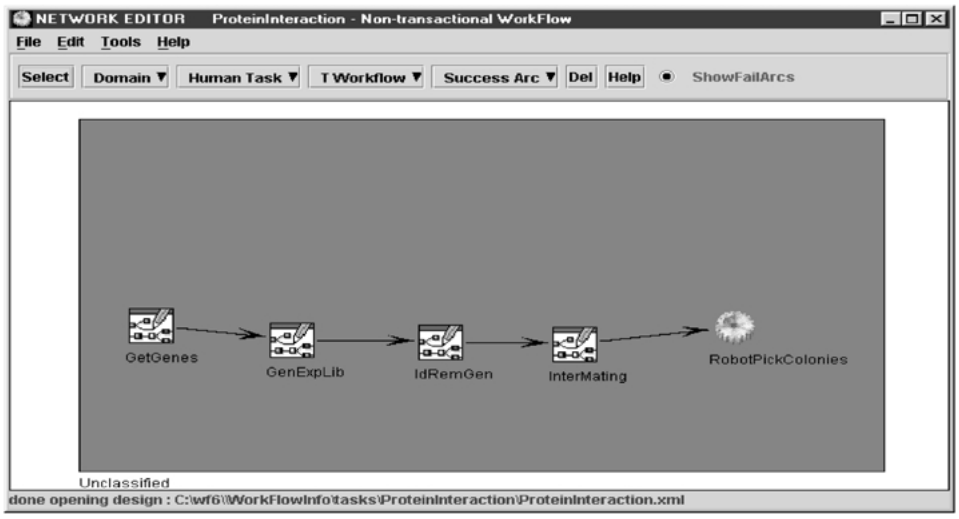

WORKFLOW DESIGNER
XML MAPPING TOOL
QUALITY METRIC SPECIFICATION TOOL
FUNGAL GENOME DATABASE CORE
QUALITY MONITOR
TASK MANAGER
TASK MANAGER
MOBILE TASK MANAGER
TASK
TASK
DB
MOBILE TASK
TASK MANAGER
TASK
WEB
INTELLIGENT ADAPTATION & SIMULATION AGENTS
METEOR-ORBWork FUNGAL GENOME PROCESS MANAGER

Figure 6. IntelliGEN architecture.


NETWORK EDITOR ProteinInteraction - Non-transactional WorkFlow
File Edit Tools Help
Select Domain Human Task T Workflow Success Arc Del Help ShowFailArcs
GetGenes
GenExpLib
IdRemGen
InterMating
RobotPickColonies
Unclassified

done opening design : C:\wf6\WorkFlowInfo\tasks\ProteinInteraction\ProteinInteraction.xml

Figure 7. Graphical design tool displaying top-level protein interaction workflow.

divided into several sub-tasks. As an illustration, the internal tasks of GetGenes are depicted in figure 8.

The GetGenes subworkflow may be initiated by obtaining genes from our cDNA libraries or from our cosmid libraries [49]. The subsequent steps include the following activities:

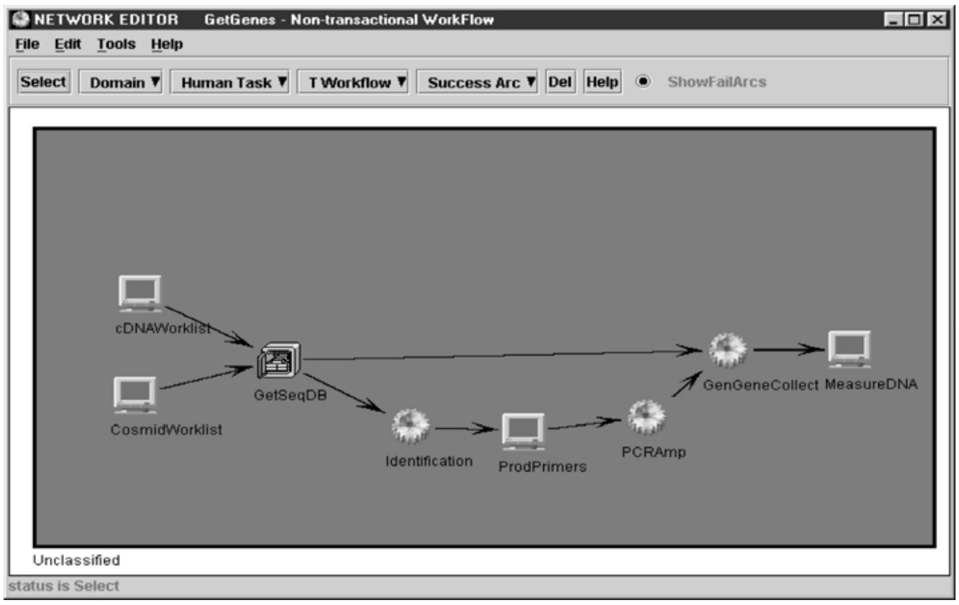


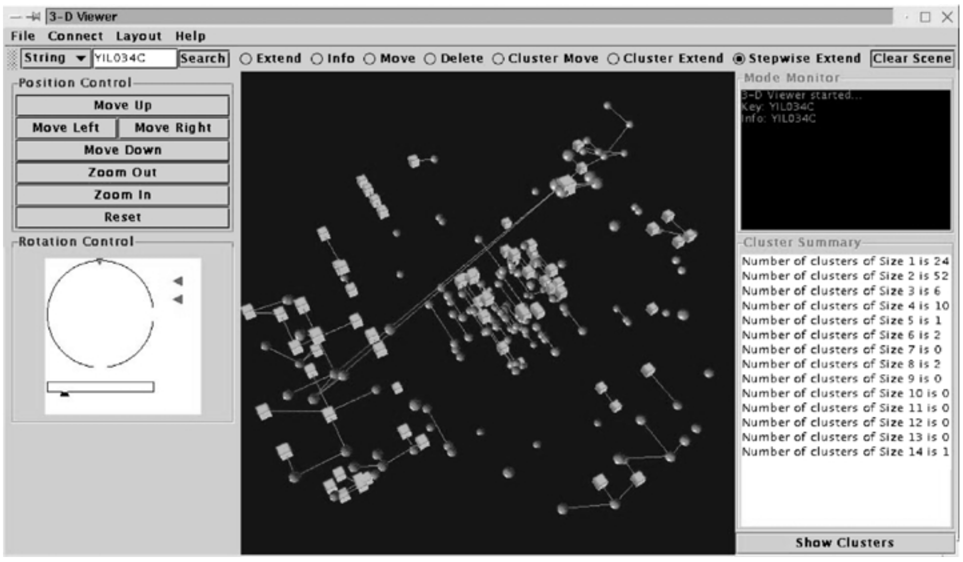
Figure 8. The details of GetGenes sub-workflow.

Clones are chosen from a library for protein-protein interaction mapping. If we elect to get genes from the cosmid libraries, genes are identified [52]. DNA is extracted from the cosmid; a particular gene amplified by PCR and recombinationally cloned into a Gateway entry vector pENTR 1A; alternatively, a cDNA collection in a Gateway entry vector is accessed, a plate pulled, and the associated genes being processed with their associated BLAST reports are displayed (not shown). All of these activities require numerous software components and a laboratory information system is used to track micro-titer plates and data, forwarding the relevant information to the follow-up tasks in the workflow. As data are tracked, an adaptive component is needed to suggest corrective action when a failure occurs during the workflow so that the throughput is sustained, or when a new interesting interaction is discovered.

In Table 2, we list some of the well-known biological software systems used in the protein interaction workflow. The individual tasks in this workflow are spread across several UNIX servers at UGA in two laboratories and a central location. The ASSEMBLY_1

Table 2. Commonly used biological software systems incorporated in IntelliGEN.

ASSEMBLY_X	Phred [25]	Base calling
	Phrap	Sequence assembly
	Consed [30]	Editing and primer design
HTG_Submission	Sequin	Submits sequence to NCBI
ANNOTATION	BLAST [4]	Sequence similarity search
	GeneMark.hmm	Gene identification



```
3-D Viewer started...
Key: YIL034C
Info: YIL034C
```

```
Number of clusters of Size 1 is 24
Number of clusters of Size 2 is 52
Number of clusters of Size 3 is 6
Number of clusters of Size 4 is 10
Number of clusters of Size 5 is 1
Number of clusters of Size 6 is 2
Number of clusters of Size 7 is 0
Number of clusters of Size 8 is 2
Number of clusters of Size 9 is 0
Number of clusters of Size 10 is 0
Number of clusters of Size 11 is 0
Number of clusters of Size 12 is 0
Number of clusters of Size 13 is 0
Number of clusters of Size 14 is 1
```

Figure 9. Screen shot of the protein-protein interaction layout tool.

and ASSEMBLY_2 sub-workflows are executed on separate servers. The annotation sub-workflow, which is computation intensive, is executed on a 32 processor SGI Origin 2000 server. The remaining tasks run on another server. Roles were created for sequence finisher, submitter, and annotator. The Web interface to METEOR creates URL's for each role with links to all tasks associated with that role.

In the final step of the workflow, in which a researcher is creating the protein-protein interaction map, we use a software system to assist the researcher in constructing the map. A screen shot of the tool (invoked automatically by IntelliGEN) is presented as an illustration in figure 9.

IntelliGEN incorporates subsystems for quality measurement and intelligent adaptation. These two novel components are briefly described in the following two sections.

4.1. Quality of service management

Workflow systems have been used to support various types of processes for more than a decade now. Workflows modeling genomic applications, such as protein interaction mapping, require the specification of Quality of Service (QoS) items such as products to be delivered, deadlines, quality of products, quality of information (e.g., accurate protein interaction maps), reliability, and cost of services [17].

Low-level script-based applications or other traditional workflow approaches do not permit flexible specification and realization of such quality requirements. Thus, they are less suitable for mission critical applications such as a protein interaction workflow. A vital goal is the ability to construct and deploy truly dependable workflows with continuous availability and predictable QoS metrics.

The management of QoS metrics directly impacts the success of genomic experiments. Therefore, when experiments are created or managed using workflows, the underlying workflow system must accept the specifications and be able to estimate, monitor, and control the QoS of processes.

For genomic laboratories, being able to characterize workflows based on QoS has four direct advantages. First, it allows laboratories to translate their experiments into their processes more efficiently, since workflow can be designed according to QoS metrics. Second, it allows for the selection and execution of workflows based on their QoS, to better follow experimental strategies. Third, it makes possible the monitoring of workflows based on QoS. QoS monitoring allows adaptation strategies to be triggered when undesired metrics are identified or threshold values violated. Fourth, it allows for the evaluation of alternative strategies when adaptation is necessary. The environment has an important impact on the strategies, methodologies, and structure of genetic workflows. Thus, in order to complete a workflow according to the initial QoS requirements, the workflow will likely be adapted, modified, and rescheduled, due to unexpected progress delays, or technical conditions. When adaptation is necessary, a set of potential alternatives is generated with the objective of changing a workflow, such as its QoS continues to meet initial requirements. For each alternative, prior to actually carrying out the adaptation, it is necessary to estimate its impact on the QoS of the modified workflow.

We have enhanced our workflow system to support processes constrained by QoS requirements, such as the protein interaction workflow. The enhancements include the development and support of a comprehensive QoS model and the implementation of methodologies (a mathematical model and simulation) to compute and predict workflow QoS. We have developed a stochastic workflow reduction algorithm (SWR) for the step-by-step computation of QoS metrics.

One of the main modules, the *Quality Monitor*, oversees the execution of the workflows and checks the quality of produced data according to QoS specifications. If the quality drops below a certain threshold the *Intelligent Adaptation* subsystem is invoked to suggest corrective actions and adapt the workflow (as described in the following subsection) so that the initial QoS requirements can be met.

The quality monitor displays QoS metrics in several formats. It is possible to color code the protein-protein interaction by whether or not a particular link is supported by two or more experiments, the number of votes for the interaction by the 4 reporters, and the number of controls satisfied in figure 5 [88]. Another possibility is the traditional quality control chart with the measures of quality on the y-axis and the course of the experiment (i.e., plate number in the high-throughput screen) on the x-axis. In the high-throughput screen, quality measures include false negative and false positive rates per plate, the average number of positive votes per positive from one bait plate, the number of controls satisfied per bait plate, and estimated coverage.

4.2. Adaptation

Traditional WfMSs are adequate to support workflows with a defined structure and with no need to account for ad hoc deviations or dynamic extensions at run-time [70]. But, recently there has been an increasing demand in developing WfMSs with dynamic capabilities, with a special emphasis to dynamic changes at the instance level. This makes sense since there are in reality very few workflows that are static (i.e. without a need to change their structures over time). As workflow processes are instantiated, changes in the environment or in previous activities may invalidate the current workflow instances, requiring adaptation procedures to be carried out. It is therefore important to be able to continuously repair or improve the execution of a workflow process [11].

A critical challenge for the IntelliGEN management system is its ability to respond effectively to changes. Changes may range from ad-hoc modifications of the process for a single experiment due to a control failing, improvement in the process to incorporate a new technique or instrument, or to a restructuring of the process to improve its efficiency. For example, it may happen that gene identification evolves to the point that gene extraction from an available cosmid library becomes feasible [41, 49] as opposed to extracting genes from a cDNA library. As a result, the run-time process used in practice is often much more variable than the process specified at design-time. It has been our experience in physical mapping and sequencing that workflows are constantly being improved. If the researchers are forced to bypass the workflow management system quite frequently, the system becomes more a liability than an asset. Thus, in ORBWork system we have implemented a layer that permits the realization of dynamic change of instances in a consistent manner [18]. The implemented module guarantee that all consistency constraints that have been ensured prior to a dynamic change are also ensured after the workflow instances have been modified [70].

When adaptation is required in the protein-protein interaction workflow, it is necessary to evaluate alternatives with the objective of changing the workflow such as its QoS continues to meet initial requirements. Adaptation procedures are evaluated based on their impact on workflow QoS that include: (1) time of execution; (2) cost of execution; (3) and quality of execution. The application of a specific adaptation procedure is constrained with the objectives set by the initial project and includes: (1) time remaining; (2) budget remaining; (3) and current measures of quality like coverage, false positive rate, and false negative rate.

Some of the changes include the type of screen, the controls to be executed, the 3AT concentration, and the type of interaction trap. To make today's workflow management systems more flexible, it is crucial to know what kinds of changes need to be supported. Changes can be triggered by developments *outside* the management system, i.e., the context/environment. There are three basic types of external circumstances that may trigger a change in the protein-protein interaction workflow: (1) *discovery* (e.g., of a new molecular machine), (2) *changing quality standards* (i.e., the change is triggered by refinements to the map of quality standards), (3) *changing technology for detecting protein-protein interactions* (i.e., due to the development of new technologies or changes in the technical infrastructure). A change can also be triggered by developments *inside* the system. These changes are not initiated by the environment, but by problems detected inside the management system itself (e.g., logical design errors or technical problems). It is important to classify the type of changes that may occur. Thus, we characterize changes as either ad-hoc or evolutionary changes:

- *Ad-hoc changes* may be the result of an error, an exception, a rare event, or special demands created by the presence of a promiscuous protein. Ad-hoc changes affect only one case (i.e., one bait plate) or a selected group of cases (a selected group of plates). They occur on an individual or selective basis. In general, it is not necessary to change the workflow definition, since the same change will likely not be needed again. An example of an ad-hoc change is the need to skip a task in case of an emergency (i.e., the reagent 5FOA was degraded or the PCR kit failed). This type of change is often initiated by some external factor. Typical problems related to ad-hoc changes are deciding what kinds of changes are allowed and the fact that it is impossible to foresee all possible ad-hoc changes.

- *Evolutionary changes* are of a structural nature. From a certain moment in time, the workflow changes for all new instances created. It is possible that the existing running instances may be influenced. An evolutionary change is applied to a workflow as a result of the adoption of a new mapping strategy, reengineering efforts, or a permanent alteration of external conditions (e.g., a change of interaction trap). Evolutionary change is typically initiated by a researcher to improve efficiency, quality or responsiveness to the community, or is forced by improved quality standards. As an example, we have two strategies for creating the protein-protein interaction map. One strategy involves the high throughput screen and the other the clone-by-clone screen. They differ on the basis of the size of the bait and prey pools being interrogated. The high-throughput screen is estimated to take 31 weeks to finish with a pool size of 96 baits vs. the whole AD-library, but it is 3-fold less sensitive than the clone-by-clone screen. In contrast, we can perform only about 650 clone-by-clone screens in a year. Once the map is sketched by the high-throughput screen, we want to turn to the clone-by-clone screen to color in the finer details of the map. Alternatively, once a human observer finds interesting clusters by the high throughput screen, more sensitive (although slower) clone-by-clone screens may be interjected to respond to community interest. Adaptive workflows can then be used to adjust the overall experimental strategy for finding protein-protein interactions using a task replacement policy.

Both ad-hoc and evolutionary changes are possible at entry time or and on-the-fly to running instances. Customizing the process definition for a single case before the processing is started corresponds to an ad-hoc change at entry time. If such a customization is also allowed after a workflow processing is started, we define it as an on-the-fly change. If evolutionary changes are only possible at entry time, then only the new cases that are started after the change took place have to run according to the updated workflow definition; all other cases run according to the old workflow definition. On-the-fly evolutionary changes are more difficult to handle since for each running workflow instance it must be decided how to deal with the change [3]. It is especially difficult to introduce changes while workflow instances are being processed. For workflow instances that are active (started, but not finished) at the time of the change the transactional tasks (possibly subworkflows) can be rolled back, and restarted under the new plan, or the instances in progress be allowed to continue under the modified workflow.

4.3. System performance

The nature of the protein-protein interaction workflow is that all of the tasks are of long duration. As an example, each of the high-throughput screens involves processing 500 "bait"

Table 3. Approximate timings for individual workflow steps.

Task	Time	Comments
Step 1	Overnight	Mix of automatic and manual (human) tasks
Step 2	1 day	A manual task
Step 3	Less than 1 day	A manual task
Step 4	4–5 days	A manual task
Step 5	1–2 days	Performed by a robot
Step 6	1 day	Largely automatic; sequence assembly and BLAST searches
Step 7	Several minutes	Human with computer assistance

library plates in a 2-hybrid screen for interactions. Unassisted (manual) processing typically takes one month per plate. Using IntelliGEN, we are able to process up to 16 plates per week, which translates to 64-fold productivity gain.

Table 3 illustrates the duration of the individual steps of in the workflow. The times are for an individual work item, which in the case of the protein-protein interactions is processing of a single plate. As easily seen, the system overhead introduced by IntelliGEN is negligible in comparison to the task processing time. Task activation time is below one second.

In addition, our estimate is that it will take approximately 75,000 task executions to complete the project, which will take roughly 3 years. This translates to about 75 tasks per day, on average, a rather small number.

5. Conclusions and future research

We have successfully applied workflow technology to a large genomic project of mapping protein-protein interactions of fungi. We have built on the success of our previous workflow project for sequencing for *N. crassa*. It is an ongoing project and changes are made to both the workflow engine, and the workflow specification, as necessitated by the laboratory work. Because of the fact that we used our distributed and dynamic enactment system ORBWork, it is possible to make changes to the deployed system. In the near future, we intend to experiment with adding to IntelliGEN agents capable of performing adaptive changes without human involvement.

The use of workflows and workflow systems to conduct and coordinate genomic experiments in a heterogeneous and distributed environment has an immediate operational requirement: the management of workflow quality of service. The experiments cannot be undertaken while ignoring the importance of quality of service measurements. When adaptation is required, an adaptation agent will use QoS information—such as the time and the cost of each task, and the final quality standards to be achieved [17]—to select a set of adaptation strategies to be applied to a workflow requiring changes. In the case that the intelligent agent is attempting to maximize coverage given the budget and time constraints, it will be necessary to invoke a simulation agent. A protein-protein interaction can be simulated as previously described [63]. The proposed strategy can be applied to many simulations of the protein-protein interaction map, and an average coverage of the networks can be estimated. In this way, different proposed workflows can be evaluated with respect to coverage.

References

1. W. Aalst and T. Basten, "Inheritance of workflows: An approach to tackling problems related to change," Computing Science Reports 99/06, Eindhoven University of Technology, Eindhoven, 1999.
2. W. Aalst and K. Hee, Workflow Management: Models, Methods, and Systems, MIT Press: Cambridge, MA, 2002.
3. W. Aalst and S. Jablonski, "Dealing with workflow change: Identification of issues and solutions," International Journal of Computer Systems, Science, and Engineering, vol. 15, no. 5, pp. 267–276, 2000.
4. S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped BLAST and PST-BLAST: A new generation of protein database search programs," Nucleic Acis Research, vol. 25, pp. 3389–3402, 1997.
5. M. Ansari, L. Ness, M. Rusinkiewicz, and A. Sheth, "Using flexible transactions to support multisystem telecommunication applications," in Proceedings of the 18th Intl. Conference on Very Large Data-bases, Aug. 1992, pp. 65–76.
6. J. Arnold, Editorial. Fungal Genetics and Biology, vol. 21, pp. 254–257, 1997.
7. J. Arnold and M.T. Cushion, "Constructing a physical map of the *Pneumocystis* genome," J. Euk. Microbiol., vol. 44, p. 8S, 1997.
8. B. Arpinar, J. Miller, and A. Sheth, "An efficient data extraction and storage utility for XML documents," 39th ACM Southeast Conference, Athens, GA, March 2001, pp. 293–295.
9. G.W. Beadle and E.L. Tatum, "Genetic control of biochemical reactions in Neurospora," in Proceedings of the National Academy of Sciences, USA, vol. 27, pp. 499–506, 1941.
10. J.W. Bennett and J. Arnold, "Genomics of fungi. The Mycota VIII," in Biology of the Fungal Cell, Howard and Gow (Eds.), Springer-Verlag: NY, 2001, pp. 267–297.
11. P.M. Berry and K.L. Myers, "Adaptive process management: An AI perspective," in ACM Conference on Computer Supported Cooperative Work, Seattle, Washington, 1998.
12. U.S. Bhalla and R. Iyengar, "Emergent properties of networks of biological signaling pathways," Science, vol. 283, pp. 381–387, 1999.
13. S.M. Bhandarkar and J. Arnold, "Parallel simulated annealing on the hypercube for chromosome reconstruction, invited paper," in Proc 14th IMACS World Congress on Computational and Applied Mathematics, Atlanta, GA, vol. 3, pp. 1109–1112, 1994.
14. S.M. Bhandarkar, S. Chirravuri, S. Machaka, and J. Arnold, "Parallel computing for chromosome reconstruction via ordering of DNA sequences," Parallel Computing, vol. 24, pp. 1177–1204, 1998.
15. S.M. Bhandarkar, S.A. Machaka, S.S. Shete, and R.N. Kota, "Parallel computation of a maximum likelihood estimator of a physical map," Genetics, vol. 157, pp. 1021–1043, 2001.
16. A.J. Bonner, A. Shrufi, and S. Rozen, "LabFlow-1: A Database benchmark for high-throughput workflow management," in Proceedings, Fifth International Conference on Extending Database Technology (EDBT), Avignon, France, March 1996, pp. 463–478. Springer-Verlag, Lecture Notes in Computer Science, vol. 1057.
17. J. Cardoso, J. Miller, and A. Sheth, "Workflow quality of service: Its specification and computation," Technical Report, LSDIS Lab, Computer Science, University of Georgia, April 2002.
18. Y. Chen, "Design and implementation of dynamic process definition modifications in OrbWork enactment system," Masters Thesis, UGA, 2000.
19. A. Cichocki and M. Rusinkiewicz, "Migrating workflows," Advances in Workflow Management Systems and Interoperability, Istanbul, Turkey, 1997.
20. A.J. Cuticchia, J. Arnold, H. Brody, and W.E. Timberlake, "CMAP: Contig mapping and analysis package: A relational database for chromosome reconstruction," CABIOS, vol. 8, pp. 467–474, 1992.
21. R.H. Davis, Neurospora Contributions of a Model Organism, Oxford University Press, New York, 2000.
22. J.L. DeRisi, V.R. Iyer, and P.O. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," Science, vol. 278, pp. 680–686, 1997.

23. L. Dogac, A. Kalinechenko, T. Ozsu, and A Sheth (Eds.), "Workflow management systems and interoperability," NATO ASI Series F, vol. 164, Springer Verlag: Berlin, 1998, p. 524.
24. C. Ellis, K. Keddara, and G. Rozenberg, "Dynamic changes within workflow systems," in Proc. of the Conf. on Organizational Computing Systems (COOCS'95), 1995.
25. B. Ewing and P. Green, "Base calling of automated sequencer traces using Phred II: Error probability," Genome Research, vol. 8, pp. 186–194, 1998.
26. X. Fang, J. Arnold, and J.A. Miller, "J3DV: A java-based 3D database visualization tool," Software—Practice and Experience, vol. 32, no. 5, pp. 443–463, 2002.
27. R.F. Geever, L. Huiet, J.A. Baum, B.M. Tyler, V.B. Patel, B.J. Rutledge, M.E. Case, and N.H. Giles, "DNA sequence, organization and regulation of the *qa* gene cluster of *Neurospora crassa*," J. Mol. Biol., vol. 207, pp. 15–34, 1989.
28. D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to infrastructure for automation," Distributed and Parallel Databases Journal, vol. 3, no. 2, pp. 119–153, 1995.
29. N. Goodman, S. Rozen, and L.D. Stein, "The labflow system for workflow management in large scale biology research laboratories," in 6th Int. Conf. on Intelligent Systems for Molecular Biology, Montreal, Canada, AAAI Press: Menlo Park, 1998, pp. 69–77.
30. D. Gordon, C. Abajian, and P. Green, "Consed: A graphical tool for sequence finishing," Genome Research, vol. 8, pp. 195–202, 1998.
31. N. Guimaraes, P. Antunes, and A. Pereira, "The integration of workflow systems and collaboration tools," Advances in Workflow Management Systems and Interoperability, Istanbul, Turkey, 1997.
32. D. Hall, "New computational tools for genome mapping," Ph.D. Dissertation, University of Georgia, 1999.
33. R.D. Hall, S. Bhandarkar, and J. Arnold, "ODS2: A multi-platform software application for creating integrated physical and genetic maps," Genetics, vol. 157, pp. 1045–1056, 2001a. Also in Hall, RD "New computational tools for genome mapping," Ph.D. Dissertation, University of Georgia, 1999.
34. R.D. Hall, J.A. Miller, J. Arnold, K.J. Kochut, A.P. Sheth, and M.J. Weise, "Using workflow to build an information management system for a geographically distributed genome sequencing initiative," in Genomics of Plants and Fungi, R.A. Prade and H.J. Bohnert (Eds.), Marcel Dekker: New York, in press.
35. D. Hall, J. Miller, M. Weise, J. Arnold, K. Kochut, and A. Sheth, "Using workflow to build an information management system for a geographically distributed genome initiative," submitted. In Hall, RD "New computational tools for genome mapping," Ph.D. Dissertation, University of Georgia, 1999.
36. Y. Han and A. Sheth, "On adaptive workflow modeling," in 4th International Conference on Information Systems Analysis and Synthesis, Orlando, Florida, July 1998.
37. C. Hensinger, M. Reichert, Th. Bauer, Th. Strzeletz, and P. Dadam, "ADEPTworkflow—Advanced workflow technology for the efficient support of adaptive, enterprise-wide processes," in Conference on Extending Database Technology, Konstanz, Germany, March 2000.
38. T. Hermann, "Workflow management systems: Ensuring organizational flexibility by possibilities of adaptation and negotiation," in Proc. of the Conf. on Organizational Computing Systems (COOCS'95), 1995.
39. D. Hollingsworth, "The Workflow Reference Model," The Workflow Management Coalition, 1994.
40. http://gene.genetics.uga.edu. Fungal Genome Resource.
41. J.R. Hudson, E.P. Dawson, K.L. Rushing, C.H. Jackson, D. Lockshon, D. Conover, C. Lanciault, J.R. Harris, S.J. Simmons, R. Rothstein, and S. Fields, "The complete set of predicted genes from *Saccharomyces cerevisiae* in a readily usable form," Genome Research, vol. 7, pp. 1169–1173, 1997.
42. C.A. Hutchison, S.N. Peterson, S.R. Gill et al., "Global transposon mutagenesis and a minimal Mycoplasma genome," Science, vol. 286, pp. 2165–2169, 1999.
43. International Human Genome Sequencing Consortium, "Initial sequencing and analysis of the human genome," Nature, vol. 409, pp. 860–918, 2001.
44. T. Ito, K. Tashiro, S. Muta, R. Ozawa, T. Chiba, M. Nishizawa, K. Yamamoto, S. Kuhara, and Y. Sakaki, "Toward a protein-protein interaction map of the budding yeast: A comprehensive system to to examine two-hybrid interactions in all possible combinations between the yeast proteins," PNAS USA, vol. 97, pp. 1143–1147, 2000.

45. S. Jablonski, K. Stein, and M. Teschke, “Experiences in workflow management for scientific computing,” in Proceedings of the Workshop on Workflow Management in Scientific and Engineering Applications (at DEXA97), Toulouse, France, 1997.
46. JDO, “Java data object expert group,” Java Data Object. 2000. JSR000012, Version 0.8. http://java.sun.com/aboutJava/communityprocess/review/jsr012/index.html.
47. J. Kececioglu, H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron, “A polyhedral approach to sequence alignment problems,” Discrete Applied Mathematics, vol. 104, pp. 143–186, 2000.
48. J.D. Kececioglu and E.W. Myers, “Combinatorial algorithms for DNA sequence assembly,” Algorithmica, vol. 13, pp. 7–51, 1995.
49. H.S. Kelkar, J. Griffith, M.E. Case, S.F. Covert, R.D. Hall, C.H. Keith, J.S. Oliver, M.J. Orbach, M.S. Sachs, J.R. Wagner, M.J. Weise, J. Wunderlich, and J. Arnold, “The *Neurospora crassa* genome: Cosmid libraries sorted by chromosome,” Genetics, vol. 157, pp. 979–990, 2001.
50. K.J. Kochut, J. Arnold, J.A. Miller, and W.D. Potter, “Design of an object-oriented database for reverse genetics,” in Proceedings, First International Conference on Intelligent Systems for Molecular Biology, L. Hunter, D. Searls, and J. Shavlik (Eds.), AAAI Press: Menlo Park, CA, 1993, pp. 234–242.
51. K.J. Kochut, A.P. Sheth, and J.A. Miller, “Optimizing workflows,” Component Strategies, vol. 1, pp. 45–57 (SIGS Publications), 1999.
52. E. Kraemer, J. Wang, J. Guo, S. Hopkins, and J. Arnold, “An analysis of gene-finding approaches for *Neurospora crassa*,” Bioinformatics, vol. 17, pp. 901–912, 2001.
53. N. Krishnakumar and A. Sheth, “Managing heterogeneous multi-system tasks to support enterprise-wide operations,” Distributed and Parallel Databases Journal, vol. 3, no. 2, 1995.
54. K. Lee, J.J. Loros, and J.C. Dunlap, “Interconnected feedback loops in the Neurospora Circadian system,” Science, vol. 289, pp. 107–110, 2000.
55. Z. Luo, A. Sheth, K. Kochut, and B. Arpinar, “Exception handling for conflict resolution in cross-organizational workflows,” Technical Report, LSDIS Lab, Computer Science, University of Georgia, April 2002.
56. Z. Luo, A. Sheth, K.J. Kochut, and J.A. Miller, “Exception handling in workflow systems,” Applied Intelligence: The International Journal of AI, Neural Networks, and Complex Problem-Solving Technologies, vol. 13, no. 2, pp. 125–147, 2000.
57. R. McClatchey, J.-M. Le Geoff, N. Baker, W. Harris, and Z. Kovacs, “A distributed workflow and product data management application for the construction of large scale scientific apparatus,” Advances in Workflow Management Systems and Interoperability, Istanbul, Turkey, 1997.
58. METEOR project home page, http://lsdis.cs.uga.edu/proj/meteor/meteor.html
59. J.A. Miller, J. Arnold, K.J. Kochut, A.J. Cuticchia, and W.D. Potter, “Query driven simulation as a tool for genetic engineers,” in Proceedings of the International Conference on Simulation in Engineering Education, Newport Beach, CA, 1991, pp. 67–72. Also at http://chief.cs.uga.edu/∼miller/papers
60. D. Miller, J. Guo, E. Kraemer, and Y. Xiong, “On-the-fly calculation and verification of consistent steering transactions,” in Proceedings of the Supercomputing Conference (SC2001), Denver, Colorado, 2001.
61. J. Miller, D. Palaniswami, A. Sheth, K. Kochut, and H. Singh, “WebWork: METEOR’s web-based workflow management system,” Journal of Intelligent Information Systems (JIIS), vol. 10, pp. 186–215, 1998.
62. J.A. Miller, A. Sheth, K.J. Kochut, and X. Wang, “CORBA-based run time architectures for workflow management systems,” Journal of Database Management, Special Issue on Multidatabases, vol. 7, no. 1, pp. 16–27, 1996.
63. J.A. Miller, A. Sheth, K.J. Kochut, X. Wang, and A. Murugan, “Simulation modeling with workflow technology,” in Proceedings of the 1995 Winter Simulation Conference, Dec. 1995, pp. 612–619. Also at http://chief.cs.uga.edu/∼miller/papers.
64. OMG 2001. OMG, UML Resources Page, http://www.omg.org/technology/uml.
65. D.D. Perkins, “*Neurospora*: The organism behind the molecular revolution,” Genetics, vol. 130, pp. 687–701, 1992.
66. D.D. Perkins, “*Neurospora crassa* genetic maps,” in Genetic Maps: Locus Maps of Complex Genomes, S.J. O’Brien (Ed.), Cold Spring Harbor Press: Cold Spring Harbor, NY, pp. 3.11–3.20, 1993.
67. D.D. Perkins, M.A. Sachs, and A. Radford, “The neuorspora compendium chromosomal loci,” Academic Press: New York.

68. D.D. Perkins, B.C. Turner, and E.G. Barry, “Strains of *Neurospora* collected from nature,” Evolution, vol. 30, pp. 281–313, 1976.
69. R.A. Prade, J. Griffith, K. Kochut, J. Arnold, and W.E. Timberlake, “*In vitro* reconstruction of the *Aspergillus*(=*Emericella*) *nidulans* genome,” in Proceedings of the National Academy of Sciences USA, vol. 94, pp. 14564–14569, 1997.
70. M. Reichert and P. Dadam, “ADEPTflex—Supporting dynamic changes of workflows without losing control,” Journal of Intelligent Information Systems—Special Issue on Workflow Managament, vol. 10, no. 2, pp. 93–129, 1998.
71. J. Rumbaugh, Ivar Jacobson, and Grady Booch, The Unified Modeling Language Reference Manual, Addison-Wesley: Reading, MA, 1998.
72. A. Sheth, “From contemporary workflow process automation to adaptive and dynamic work activity coordination and collaboration,” in Proceedings of the Workshop on Workflows in Scientific and Engineering Applications, Toulouse, France, 1997.
73. A. Sheth, W. Aalst, and I. Arpinar, “Processes driving the networked economy,” IEEE Concurrency, vol. 7, no. 3, pp. 18–31, 1999.
74. A. Sheth and K.J. Kochut, “Workflow applications to research agenda: Scalable and dynamic work coordination and collaboration systems,” Workflow Management Systems and Interoperability, A. Dogac et al. (Eds.), Springer Verlag: Berlin, 1998, pp. 35–60.
75. A. Sheth, K.J. Kochut, J.A. Miller, D. Worah, S. Das, D. Lin, D. Pallaniswami, J. Lynch, and I. Shevchenko, “Supporting state-wide immunization tracking using multi-paradigm workflow technology,” in Proceedings of the 22nd International Conference on Very Large Data Bases, Bombay, India, 1996, pp. 263–273.
76. A. Sheth, D. Worah, K.J. Kochut, J.A. Miller, K.E. Zheng, D. Palaniswami, and S. Das, “The METEOR workflow management system and its use in prototyping significant healthcare applications,” in Proceedings Toward an Electronic Patient Record Conference (TEPR ’97), vol. 2, Nashville, TN, 1997, pp. 267–278.
77. J. Skolnick, J.S. Fetrow, and A. Kolinski, “Structural genomics and its importance for gene function analysis,” Nature Biotechnology, vol. 18, pp. 283–287, 2000.
78. S.H. Strogatz, “Exploring complex networks,” Nature, vol. 410, pp. 268–276, 2001.
79. Tian, Hui, “Storage management issues for high performance database visualization,” in Proceedings of the 39th Annual Southeastern ACM Conference, Athens, Georgia, March 2001, pp. 251–256.
80. P. Uetz, L. Glot, G. Cagney, T.A. Mansfield, R.S. Judson, J.R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, B. Godwin, D. Conover, T. Kalbfleish, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J.M. Rothberg, “A comprehensive analysis of protein-protein interactions in *Sacharomyces cerevisiae*,” Nature, vol. 403, pp. 623–627, 2001.
81. J.C. Venter, M.D. Adams, and E.W. Myers et al., “The sequence of the human genome,” Science, vol. 291, pp. 13040–1351, 2001.
82. M. Vidal, “Protein-protein interactions,” Encyclopedia of Genetics, Academic Press, vol. 3, pp. 1551–1552, 2002.
83. R.T. Watson, G.M. Zinkhan, and L.F. Pitt, “Object-orientation: A new perspective on strategy,” Paper read at Academic Industry Working Conference on Research Challenges, April 27–29, 2000 at Buffalo, NY.
84. D.Worah, A. Sheth, K. Kochut, and J. Miller, “An error handling framework for the ORBWork workflow enactment service of METEOR,” Technical Report, LSDIS Lab. Department of Computer Science, University of Georgia.
85. Workflow Management Coalition Standards, http://www.aiim.org/wfmc/mainframe.htm
86. S. Wu, A. Sheth, J.A. Miller, and Z. Luo, “Authorization and access control of application data in workflow systems,” Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies (JIIS), vol. 18, no. 1, pp. 71–94, 2002.
87. Z. Xu, B. Lance, C. Vargas, B. Arpinar, S. Bhandarkar, E. Kraemer, K. Kochut. J. Miller, J. Wagner, M. Weise, J. Wunderlich, J. Stringer, G. Smulian, M. Cushion, and J. Arnold, “Mapping by sequencing the Pneumocystis genome using the ODS3 tool,” Genetics, in press.
88. Y. Zhang, “A visualization system for protein interaction mapping using Java 3D technology,” Masters Thesis, UGA, 2001.