Quality of Service and Scientific Workflows*

Mladen A. Vouk
Software Engineering and Multimedia Laboratories
Department of Computer Science, Box 8206
North Carolina State University, Raleigh, NC 27695
Tel: 919-515-7886, FAX: 919-515-6497, e-mail: vouk@adm.csc.ncsu.edu

and

Munindar P. Singh
Database Laboratory
Department of Computer Science, Box 8206
North Carolina State University, Raleigh, NC 27695
Tel: 919-515-5677, FAX: 919-515-7896, e-mail: singh@adm.csc.ncsu.edu

(To be presented at the IFIP/TC2/WG 2.5 Working Conference 7 - "Quality of Numerical Software: Assessment and Enhancements," Oxford, U.K., 8-12 July 1996. Revised version will appear in the WoCo7 proceedings book that will be published by Chapman & Hall 1996).

Key Words
quality of service, QoS, problem solving environment, PSE, scientific workflow, reliability, availability, intra- and inter-net response delays, users.

Abstract

This paper discusses some quality issues that face the next generation of the **network-based** scientific problem-solving environments. The advent of high-performance computing engines and networks is opening a fantastic opportunity for bringing serious numerical and problem-solving applications closer to a broad base of potential users with widely differing needs. From the perspective of these users, a key issue will be the **quality of service** (QoS) in the broader sense, i.e., not just parameters such as network delays and throughput, but also end-user quality factors such as system availability, system functionality and content quality, and so on. In order to facilitate integration of the QoS and PSE concepts we view PSEs through the prism of **scientific workflows**, i.e., a series of structured activities and computations that arise in scientific problem-solving. Scientific workflows are expected to coexist and cooperate with other user workflows (e.g., business workflows, educational workflows, legislative workflows). As such they must support compatible QoS. We are using data from existing network-based systems and workflows to quantitatively bound some of the PSE QoS parameters. For example, system availability must be 0.95 or better, while adequate synchronous user-machine interactions require user-level end-to-end (round-trip) delay times that are consistently less than about 250 msec, and network-level round-trip delays that do not exceed 100 to 150 msec. Use of multimedia imposes additional restrictions, while end-user risks impose bounds on the security and reliability of numerical computations and algorithms. It is our belief that the next generation of PSEs must have QoS parameters designed into the system, or these PSEs will fail to live up to user needs and expectations.

* Research supported in part by NSF (ACS 9418960 and ACS 9696131), NASA (NAG-1-983), EPA/MCNC (94-7050-027/5-30012 and 94-7050-029/5-30816), IBM Canada (CAS) Fellowship, IBM Corp. (RTP), and CACC (61-0103/5-30456 and 83-0062/5-30781).

1. Introduction

Modern problem solving environments (PSE) are envisioned as collections of cooperating programs, tools, clients, and intelligent agents [Gal94]. These components are integrated into an environment that facilitates user interaction (such as problem statement and solution engineering) and cooperative execution of the components charged with the solution tasks. An example is a system that would help an environmental scientist or a regulator to pose environmental engineering questions (problems), develop, execute and validate solutions, analyze results, and arrive at a decision (e.g., cost-effective emission control strategy). Such a PSE would consist of a management, analysis and computational framework that would be populated with a variety of models and data that describe the science behind the phenomena, the solutions of interest and the decision rules [e.g., Den96]. It is usually assumed that a modern PSE is distributed across a number of central processing units that may or may not reside in one physical computer. In fact, the advent of high-performance computing engines and networks, the potential of new technologies (such as the Asynchronous Transfer Mode) to "guarantee quality of service", and the ready access to network-based information through the World-Wide Web (WWW) is opening a fantastic opportunity for bringing serious numerical and problem-solving applications closer to a broad base of potential users.

An important part of modern PSE framework is its ability to facilitate effective and efficient communication among the PSE components (or objects). This is recognized by both researchers and software manufacturers, and, in recent years, it has resulted in a proliferation of communication building blocks for distributed scientific computing. Two examples are PVM and MPI [Gei94, Sni95], communication libraries and message and process brokers that allow distribution of a parallelized problem over a number of processing units in order to increase the system's computational performance. Although not originally intended for this purpose, both PVM and MPI can be used to distribute not only fine-granularity solution elements (such as code segments) but whole programs and PSE parts. Another example is a variety of, usually CORBA-compliant, commercial object brokers that can be used to construct PSEs.

From the perspective of a PSE user, one of the key issues will be the **quality of service** (QoS). In this context we broaden the classical definition of QoS to include not only network-based parameters (such as response delay and throughput), but also measurable end-user quality characteristics such as system availability, performance, algorithmic scalability, effectiveness, quality of system content, quality of user-system interactions, and so on. Furthermore, in order to facilitate integration of the QoS and PSE concepts, and naturally introduce already existing formal specification and quality analysis approaches, we view PSEs as computer and network-based systems that support **scientific workflows**, i.e., a series of structured activities and computations that arise in scientific problem-solving. Scientific workflows are expected to coexist, cooperate and even meld with other user workflows (e.g., business workflows, educational workflows, legislative workflows). As such they must support compatible QoS. We can use data from existing network-based systems and workflows to quantitatively bound some of the PSE QoS parameters.

Section 2 defines the workflow view of problem solving. Section 3 discusses the quality of services issues and provides quantitative bounds for some more prominent QoS parameters. Conclusions and summary are given in Section 4.

2. Problem Solving and Scientific Workflows

Much progress has recently made in high-performance problem-solving environments [e.g., Gal94]. Such environments often involve complex, structured, heterogeneous, long-lived computations.

For example [Coa96], during modeling to support the North Carolina[1] CAAA State Implementation Plan (SIP), over 50 control strategies were studied in an attempt to remediate North Carolina's air problems. The UAMGUIDES[2] system was used to provide semi-automated program, data, and visualization management for this study. Under UAMGUIDES, visualization and modeling programs interact with files on the basis of file type, to help prevent errors. The SIP studies included four ozone episodes, each 3 days long with each "model run" requiring the execution of 3 one-day Urban Airshed Model (UAM) computations. One such computation includes meteorology and emissions (EPS) preprocessing as well as data postprocessing and analysis. Because of restrictions in EPS, each day's emissions simulation requires the splitting of the emissions inventory into a number of pieces (each of which it treated somewhat differently), running the programs in EPS on each piece, and then finally merging all the pieces together, for a total of more than 40 program executions per day simulated.

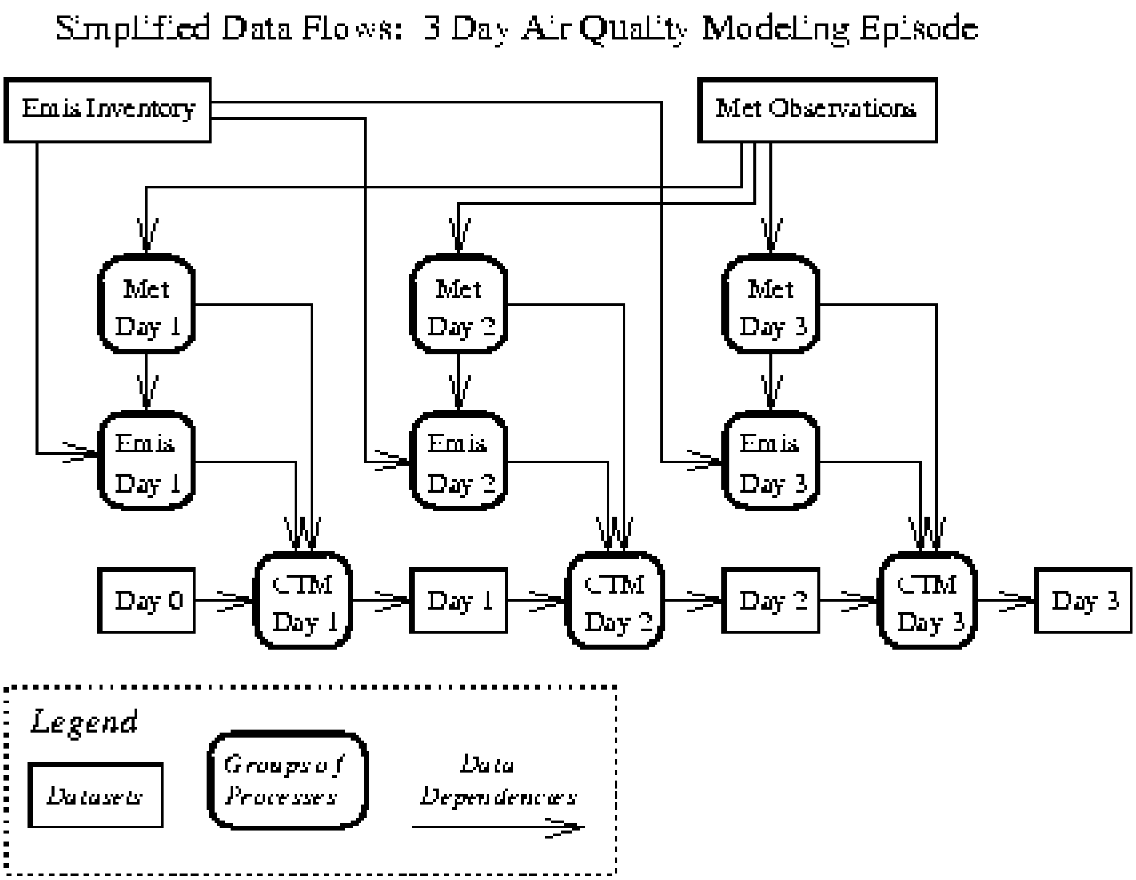

Figure 1. Data Flow for a 3 day air quality modeling episode

---

[1]Note that this is a "small" problem when compared to, for example, California or New England.
[2](see http://www.iceis.mcnc.org/uamguides/)

This workflow can be described using a directed graph, or a similar device. Figure 1 is a simplified illustration of a data-flow graph for just one 3-day episode for one control strategy. Each bubble represents a subgraph of processes[3], each of which needs to be assigned to a specific computer, and each of which needs a set of files and control parameters when it executes. In all, the study resulted in about 20,000 program runs that generate over 20,000 files that needed to be moved among the computing hosts, analyzed, indexed, managed, and archived. Each day of each simulated control strategy generates about 3 gigabytes of data, for a total data volume that exceeds 1.5 terabytes. If written down as text, the complete processing specification would probably take at least 100,000 lines—over 1500 pages—too long and too detailed for effective human inspection.

It is obvious that this type of scientific problem solving may require many different computing and networking services, and although significant advances have been made, major bottlenecks still remain in specifying and managing these computations, and in enforcing of intricate dependencies among their components [e.g., Amb95, Den96]. Appropriate abstractions can remove these bottlenecks, enable significant productivity gains and help integrate the scientific computing into general problem solving and decision-support framework.

2.1 Workflows

Interestingly, related abstractions have been intensively studied under the rubric of *workflows*. Workflows, especially those with transactional components, have drawn an enormous amount of attention in the databases and information systems research and development communities [Elm92], [Geo95], [Hsu93]. Considerable progress has been made both in workflow specification and scheduling systems based on agents, temporal logic, and process algebra. Over 100 (and according to some, 250 [Wai96]) workflow products of various shapes and sizes now exist. Much of the recent research interest in workflows has been focused on workflows as they arise in business environments, e.g., [Ell79]. The products too are geared to enterprise computing, e.g., [Ley93], enabling, for example, for various routine office activities to be captured and automated over functionality such as email. Realizing that office-work workflows are inherently limited, there has also been considerable effort in the area of workflows involving transactions and tasks that execute in heterogeneous information systems, typically those consisting of mainframe computers with arcane interfaces [Sin94]. Whereas most products have a simplistic and centralized execution engine, the legacy workflows are more heavy-duty. For example, there has been some work on making the workflow engines fault-tolerant.

Viewed in a broader sense, workflows are the natural outcome of the limitations of the traditional transactional model in capturing the semantics of activities in open, heterogeneous settings. By definition, traditional transactions are *ACID* [Gra93], which means they have the following properties:

1. Atomicity: all or none of a transaction happens
2. Consistency: a transaction takes a database from one consistent state to another
3. Isolation: the intermediate results of transactions are never visible
4. Durability: the effects of a successful transaction are permanent,

These are useful properties and responsible for the success of the traditional transactional approach. But they have some inherent limitations. In a distributed, heterogeneous environment, to guarantee atomicity requires some kind of a mutual commit protocol, which can be expensive (and sometimes impossible) to execute. Isolation requires long-lived locks on shared resources and by definition precludes cooperation. This has led to much work on the so-called extended or relaxed transaction models, which relax the ACID properties in different ways [Elm92]. The workflow paradigm has grown out of this literature (accommodating general tasks, not just

[3]There are 37 processes within each emissions bubble!

transactions), to find use in many areas. Two examples are the office-work community (enabling human collaboration), and the general process modeling community (enabling the capture of application-specific semantics of the activities of interest).

Interestingly, scientific problem-solving environments share many of the characteristics of applications in which workflows exist. This motivates us to look closer to see what synergies might be exploited in describing and managing scientific computations and the quality of service they require.

2.2 Scientific Workflows

Although business workflows deserve the attention they are receiving, another class of workflows emerges naturally in sophisticated scientific problem-solving environments. We believe this class of workflows, which we dub **scientific workflows**, will become ever more important as computing expands into the routine activities of scientists [Sin96a, Sin96b, Wai96].

We define scientific workflows as a blanket concept to describe series of structured activities and computations that arise in scientific problem-solving. These structured activities are often termed studies or experiments, and different graph-based notations, e.g., generalized activity networks (GAN) and Petri-nets, are frequently a natural way of representing the flow of numerical and human processing that takes place [Den96, Amb95, EBV95, Elm66]. These flows bear the following similarities to workflows:

1. Scientific problem-solving usually involves the invocation of a number and variety of analysis tools. However, these are typically invoked in a routine manner. For example, the computations involve much detail (e.g., sequences of format translations that ensure that the tools can process each other's outputs), and often routine verification and validation of the data and the outputs. As scientific data sets are consumed and generated by the pre- and post-processors and simulation programs, the intermediate results are checked for consistency and validated to ensure that the computation as a whole remains on track. Control and data flows cannot always be hardwired.
2. Semantic mismatches among the databases and the analysis tools must be handled, and their performance characteristics matched. Some of the tools are designed for performing simulations under different circumstances or assumptions, which must be accommodated to prevent spurious results. Heterogeneous databases are extensively accessed; they also provide repositories for intermediate results.
3. When the computation runs into trouble, error recovery is through *semantic rollforward*; just as for business workflows, *rollback* is often not an option. Rollback (or backward recovery) is the traditional paradigm for recovery in ACID transactions, and it is a very common **software fault-tolerance** mechanism [Lyu95, Lyu96, McA96]. What rollback means is that when an ongoing transaction fails, its effects are obliterated from the system. Rollback simplifies the processing and data structures required for recovery, but it is often quite inconvenient. This is because an ongoing transaction (that may have to rollback) cannot have any permanent effects, such as printing out a result. Rollback also entails that the partial results of the rolled back transaction are lost. For long-lived, complex activities, that must often interact with users, rollback is not an option.

   By contrast, rollforward (or forward recovery) means that when the given activity fails, the system picks up the pieces and automatically restarts the activity [Lyu95, Lyu96, McA96]. This entails that the system maintain considerably complex data structures. Whereas rollback can be syntactic, i.e., independent of application semantics if the transactions are perfectly isolated, rollforward typically requires knowledge of application semantics. Thus it is more complex and requires increased input from the programmer. An interesting proposal in this regard is the Contracts Model, which takes on several operating system responsibilities in order to ensure rollforward [Wac92]. We view rollforward as an instance of software fault-tolerance. This is because rollforward is a means of ensuring that a given activity completes successfully despite failures of various kinds. In this sense, it is a means to achieving a qualitatively higher service for the user. Considerable attention has been paid in the extended transaction models and workflows communities to the semantics of activities and how different sub activities may interact. These can be leveraged for scientific computing through the adoption of the scientific workflows paradigm.

4. Many large-scale scientific computations of interest are long-term, easily lasting weeks if not months [Amb95, Den96]. They can also involve much human intervention. This is especially frequent during the early stages of process (workflow) design. However, as they are debugged, the exceptions that arise are handled automatically. Thus, in the end, the production runs frequently require no more than semiskilled human support. The roles of the participating humans involved must be explicitly represented to enable effective intervention by the right person.
5. The computing environments are heterogeneous. They include supercomputers as well as networks of workstations and supercomputers. This puts additional stress on the run-time support and management, especially in the area of load balancing and the quality of service offered by the diverse hosts and interconnecting networks. Also, users typically want some kind of a predictability of the time it would take for a given computation to complete. Making estimates of this kind is extremely complex and requires performance modeling of both computational units and interconnecting networks.

Many trends observed nowadays in the scientific problem solving arena suggest that the quality of scientific problem solving that end-users **expect** requires not only provision of high quality numerical computing algorithms and software, but also integration of these solutions with advanced computational and networking frameworks, and with day-to-day operational environments and workflows within which the users exist (e.g., industrial, government or academic settings). The workflow paradigm enables appropriate description and analysis of coexistence and melding of scientific workflows with other workflows into which they have to fit, and of certain quality constraints dictated by these larger workflows. By describing problem solving as workflows, we hope to bring to bear on them the advanced techniques being developed in workflows research, as well as performance and QoS work deriving from the high-performance networking and communications arena. These include sophisticated notions of workflow specification and of toolkits and environments for describing and managing workflows, and recognition of performance issues that have previously been reserved for analysis of networks (e.g., that end-to-end throughput observed by a user at the application level will typically be that of the device or process with the lowest throughput capacity in the path). In this way, scientific workflows are to problem-solving environments what business workflows are to enterprise integration.

However, while considerable progress has been made both in i) the implementation of complex systems of scientific computations, and ii) workflow specification and scheduling, there is currently no unified theory or system that formalizes scientific workflows as defined above.

3. Some Quality (of Service) Issues

From a networking perspective the QoS is defined by measures such as *response delays, probability of loss of data, jitter, and throughput.* Since advanced network-based scientific computing and problem solving is highly dependent upon the successful performance of QoS-sensitive multimedia applications, an issue is how to guarantee QoS requirements imposed by varying mixes of transmitted voice, video, image and data that support the application. For instance, PSEs must deal with **interoperability** problems that arise between the various local-area and wide-area network guarantee mechanisms, and with the interoperability among its distributed components. In fact, in order to achieve end-user QoS guarantees, it is necessary that all interacting end-to-end entities have agreement upon the interfaces for QoS specification, for exchange of information regarding QoS requests and provisions, and for evaluation of QoS performance.

In the context of a modern network-based PSEs we broaden the classical definition of QoS to include system **reliability** and **availability**, the **adequacy** and **scalability** of the functionalities offered by the system, the quality of the information and content (e.g., models, data, educational material) that the system delivers, system efficiency and effectiveness, system **security**, and the quality of user-system interactions.

Furthermore, a good quality scientific workflow support system should have resource **adaptation capabilities** (as well as other forms of flexibility) that minimize the impact of resource limitations on the user. For example, the system would recognize limitations of a user terminal, such as lack of audio capabilities, and would automatically downgrade the information transfer mode to display the textual transcript of the audio record. Similarly the system might recognize user knowledge or experience profile and adapt its interfaces, and/or the amount and level of interaction(advice) and material it offers to the user, to suit. The necessity of serving many users, with very different desktop facilities, over highly heterogeneous networks raises a number of research and engineering problems. The issues include:

1) definition of required end-user quality of service (QoS) that such applications need to furnish;
2) development of robust QoS-sensitive interfaces, algorithms and data formats that are needed for adequate desktop delivery of functions such as animation, visualization or user-to-user collaboration;
3) development of truly distributed and responsive system elements that provide wide-area agent facilitated data queries, access to very large distributed archives, and easy use of complex numerical, statistical and visualization techniques from a desktop virtual environment; and
4) development of a network-based framework for appropriate and timely user assistance (on-line help), education and training in the area of the PSE specialty.

In the rest of this section we addresses some characteristics of network-based PSEs that we believe play a prominent role in the defining (both qualitatively and quantitatively) the "quality" of services a next-generation environment for support of scientific workflows should provide. We use our experiences with modern educational and scientific systems and workflows to provide some quantitative bounds on the QoS parameters that PSEs will need to meet.

3.1 User Diversity and System Functionality

In a PSE, the most important system entity, and the principal quality driver and constraining influence is, of course, the user. PSE users can be classified into a number of categories, of which the three most important (non-exclusive) are system developers, authors, and research & production (R&P) users. Many system requirements derive directly from the principal user profiles.

3.1.1 Users

System developers are responsible for the development and maintenance of the system framework. They develop and integrate system interfaces, administration and management software, communications and scheduling algorithms, (authoring) tools of different kinds, content access algorithms and software, and so on. They must be experts in specialized areas such as AI, databases, multimedia, software and computer engineering, and communications. They require specialized tools for system framework development, maintenance, testing and performance evaluation.

Authors are developers of the PSE-specific content. They are responsible for the development of individual algorithms, lessons, applications and solutions that are integrated into scientific problem-solving workflows by R&P users. It is essential that authors be content experts, but they should not have to be system experts. Therefore, it is important that the system *authoring tools* and interfaces are easy-to-learn and easy-to-use, and that they allow the authors to concentrate on the content development rather than struggle with the system intricacies. Some of the functionalities that a PSE framework must provide are various editors, compilers, interpreters, authoring languages, tools, the capability to gather information about the use of their solutions and about any problems encountered, and material security and recovery (including protection of copyrights, protection from system crashes and losses, etc.).

Research & production users are, of course, the most important users of the system. This category covers a broad spectrum. In the future, PSE users are likely to range from very sophisticated to very naive, from academic educators to high-school children, from individual researchers to technicians running routine analyses [e.g., Amb95, Den96]. However, they will all require appropriately reliable and timely delivery of the interaction results, easy-to-use interfaces, collaborative support for local and remote joint projects, help, and so on. Users may decide to use “canned” solutions, they may sample and combine existing and customize them, they may update existing studies or they may develop new studies. However, one thing is certain: the users will NOT use a PSE in isolation. They are likely to import and export data and results to and from other workflows, and they will expect a PSE to cooperate and coexist with other computer-based support systems. They may also teach and tutor using PSE facilities. R&P users have to be knowledgeable in the PSE application area, but their level of expertise may vary. The problem-solving system needs to adapt to their expertise level, provide advice and help, and possibly tutor them in areas where they lack knowledge. Users' work must be secure and protected from data losses and unauthorized access. The system must handle user-related information with special care. On the one hand, it must preserve the user's right to privacy, but on the other hand, it must provide the system developers and authors with enough feedback to allow them to proactively improve the system and thus better serve the users.

It is obvious that different user categories have different, and sometimes contradictory needs, and a "good quality" PSE must adequately support different user modes. Furthermore, it is very likely that the same persons may use a PSE in different roles, so the system must be capable of distinguishing and separating these different user "personalities".

For example, scientific workflows often begin as **research** workflows and end up as **production** workflows. A PSE should support and enable this transition. Early in the workflow lifecycle, there may be a need for considerable human intervention and collaboration; later workflows begin to be executed increasingly automatically. Thus in the production mode, there is typically less room for collaboration at the scientific level and the computations are more long-lived. This happens partly because of limitations of the available technology. We speculate that if true workflow technology were available to manage scientific computations, there would be a reduced push to automate everything and the quality of the solutions obtained could be improved by involving the right humans at the appropriate places. Be that as it may, during the research phase, scientific workflows need to be enacted and animated far more intensively than business workflows. In this phase, which is more extensive than the corresponding phase for business workflows, the emphasis is on execution with a view to design, and thus naturally includes iterative execution. The corresponding activity can be viewed as a correlate of business process engineering. For this reason, the approaches for constructing, managing, and coordinating process models will find useful application in scientific settings, if only the main problems are cast appropriately.

3.1.2 Functionalities

We identify two classes of user-oriented functionalities and issues that a "good quality" scientific workflow support system must be able to handle, in addition to the application-area specific knowledge, algorithms and solutions.

The first category applies to workflows in general and it includes:
i) handling exceptions and providing fault-tolerance,
ii) handling a range of user capabilities, the roles of different participants, and allowing the role bindings to change,
iii) declaratively specifying control and data flows,
iv) automatically executing and monitoring of workflows to meet stated specifications.
v) incorporating human decisions into the process, and
vi) coordinating and synchronizing with other scientific and business workflows.

The second category is specific to scientific workflows and includes the features required for scientific computations, but which may not be adequately addressed in traditional workflows. This category includes issues such as
i) relative uniqueness of each workflow, particularly during the research phase when there is less opportunity to use canned or "normal" solutions,
ii) the ability to handle a vast number and variety of analysis tools, and interfacing to a diverse array of computational environments (including clusters of networked workstations and supercomputers), and
iii) auditability of the computations when their results are used to make decisions that carry regulatory or legislative implications.

The presence (or lack) of most of these functionalities is quantifiable through system parameters such as reliability, availability, efficiency, effectiveness, productivity, cost of maintenance, etc. For example, exception handling and fault-tolerance reflect in system reliability and availability. A good workflow support environment provides for quantitative evaluation of its quality traits through capture of appropriate metrics and through utility tools that allow analysis and evaluation of the system quality parameters. A excellent example of such a built-in facility is the system reliability interface provided by an established computer-based education system called NovaNET[4].

Although interest in the quality of numerical software has been around for a long time [e.g., How80], and many very reliable numerical software libraries [e.g., NAG, IMSL] and PSEs [e.g., MATHLAB, SAS] are available, there are indications that, in general, there is considerable variance in the quality of numerical software. In fact, bounds on acceptable QoS parameters for PSEs are either non-existent, or are still more qualitative than quantitative. Since in the future scientific workflows, and by implication workflows that involve numerical computations, will need to at least match that of other workflows with which they are expected to interact and fuse, we can use existing information about some of these other workflows to estimate user-acceptable bounds for PSEs. In the following subsections we illustrate this using the data from NovaNET, and from EDSS[5] [Amb95] to address three very prominent QoS needs: system availability, system throughput, and end-to-end delays.

---

[4]NovaNET is a successful low-overhead high-yield multimedia educational system that originates from the University of Illinois at Urbana-Champaign (UIUC) and serves thousands of users on a daily basis. The NovaNET system reliability and recovery measurements are collected, processed and reported automatically and continuously, and system outages (or failures) include everything, from application software, through system hardware, software and network problems, to problems caused by operator errors.

[5]The Environmental Decision Support System (EDSS) is an experimental system being constructed by MCNC North Carolina Supercomputing Center (NCSC) in collaboration with NC State. The system provides access to

3.2 Reliability and Availability

In addition to adequate system functionality and usability, a successful scientific workflow support system must have an adequate availability, or a broad base of users will simply not use it. Availability is defined as the probability that a system will be available at any random time, t, during it's operational life [Cra92, Jon96, Dik96]. This implies appropriate system reliability and recovery rates [Jon96]. What are they? If we assume that a PSE user will be at least as discriminating and demanding as university students and educators, we can use the NovaNET data to set a lower bound on the minimally acceptable overall system availability. If we assume that a network-based system will be limited by the reliability of its network links, we can use the information on the quality of Internet switching elements [e.g., Jon96] to establish another type of bound. Of course, this assumes that numerical components of the system, individually and in combination, have sufficiently high reliability that they are not the limiting factor. In the cases where a PSE is used to make critical or high-risk decisions, the reliability of the numerical elements used in the computations that lead to the critical decisions may the governing influence since their reliability should at least match the decision risk levels [Boe89]. However, in general, networks and user interfaces may play an equally important role. For example, based on our experiences and measurements with other systems in the area of network-based multimedia, we estimate that (before error correction) acceptable network-level packet loss rate is in the range 0.02 to 0.1 for voice and audio interactions, 10^{-9} to 10^{-5} for images, zero to 10^{-5} for data, and 10^{-10} to 10^{-8} for full-motion MPEG video.

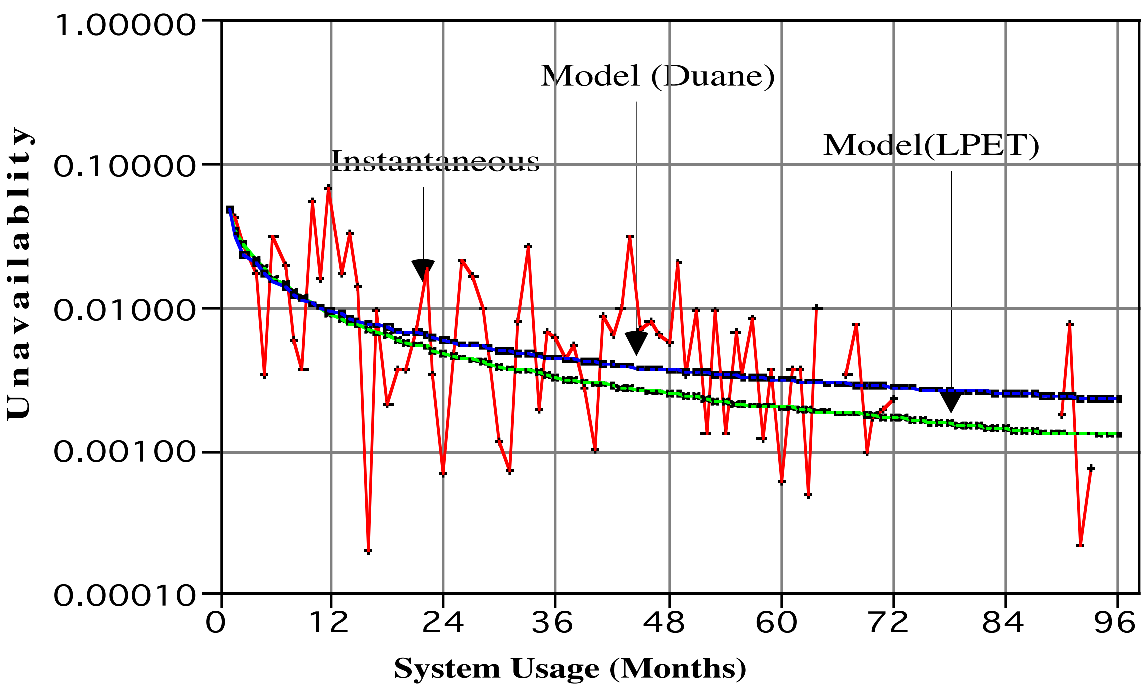

Figure 2. NovaNET system unavailability during regular operation - Actual and Models

heterogeneous database systems. A environmental study is modeled as a partial order of program invocations and (possibly) human interventions. The partial order describes the flow of data from one program to the next. Each program performs some useful function, such as a simulation, visualization, or data reduction, and consumes and produces scientific and other data sets. Although an EDSS prototype is in alpha-testing phase, full system will require integration of scientific and appropriate economic and business models and workflows so that regulatory decisions can be fully qualified.

Part of a recent NovaNET availability study [Dik96] is shown in Figure 2. The graph depicts the average unavailability of the NovaNET from 1988 to 1995 inclusive (unavailability is 1 minus availability). The data include all events during which the system was either **completely** or **partially** unavailable[6]. While most conventional availability analyses partition the observed system space into **operational** (UP) and **in-repair** (DOWN) states, the classification into many down-states is an alternative (e.g., levels of degraded performance) [Cra92]. We see that unavailability decays (availability grows) as the system matures. NovaNET system measurements indicate that, once a user starts one hour of work (e.g., a lesson), to maintain reasonable user satisfaction, the probability of getting through that hour without any problems should be above 0.95, while probabilities below 0.86 are totally unacceptable. For instance, in the April 1996 the probability of a NovaNET clock hour having no problems at all was 0.993, and the scheduled user time was error-free 99.91% of the time [Nov96]. We expect that a good PSE would not have reliability and availability characteristics that at least match above figures.

On the other hand, according to Bellcore [Bel89, Bel90a, Bel90b], public network switching elements are expected to assure unavailability that does not exceed about 10^{-5} (about 3 min. of downtime per year). Therefore, it would not be unreasonable to require that individual PSE elements provide reliability and error control (including exception handling, fault-tolerance, and graceful error trapping) at least at that level, and that the overall PSE availability during its posted user access hours be at least 0.95 (this includes everything: network outages, violation of end-to-end response times, PSE system and content software failures due to algorithmic or other problems, and so on).

3.3 Bandwidth and Delays

One the most important PSE QoS drivers is the quality of its computer-human interface (CHI). A user response to a PSE, and user's capability to start and understand the interactions, and absorb results, is a very strong function of its CHI. To achieve effective information transfer rates[7], we may need to use different sets of "symbols" and presentation rates - from simple characters (at several thousand bits per second), to sophisticated high-definition animations and full-motion movies (at many megabits per second).

3.3.1 Throughput

PSE bandwidth requirements may vary widely. From few kilobits per second to hundreds of megabits per second. Each mode of operation of a network-based PSE has certain throughput requirements. In some cases the bandwidth needs to be provided synchronously (user waits for output), and in some cases asynchronously (batch mode), both with varying delay requirements. The principal driver in deciding what is appropriate is the problem solving workflow. In a computer-based PSE it usually takes one of the two forms:

(1) "TV-model" format; This is a high average bandwidth synchronous (real-time) full-motion audio/video interaction that can be found in video-conferencing and video-based collaborative work, or a large-scale real-time data acquisition effort; These exchanges can be very "bandwidth-hungry" requiring between 6 and 45 megabits per second (Mbps) per session, depending on the compression mode used and the desired quality of service. Large-scale real-time data collection, such as that occurring in some medical PSEs, can be even more demanding (throughput requirements can be as high as 400 Mbps).

---

[6] We are in the process of recovering the data for months 73 thru 91 from backup tapes.

[7] Research shows that humans cannot extract (reason about, learn) new information at the rate faster than about 20 bits/second (i.e., differentiate among about 1,000,000 "symbols" each second) [Hal77, Str67, Qua55].

(2) "Data-model" format, i.e., low to medium average bandwidth synchronous (real-time) interactions with asynchronous data transfers. In this format one attempts to judicious use of hypertext, animation, graphics, voice and text to adaptively deliver the material. Hence, *synchronous* interactions in this format may be very bursty. The average required throughput may be quite low, in the range 1000 to 20,000 bits per second, but because the throughput needs can vary widely, peaks can be as high as 100 Kbits/s to 2 Mbits/s. Thus, real-time bandwidth-on-demand is a network feature that can greatly enhance this mode of interaction [Rin95]. *Asynchronous* transfers may require an even larger bandwidth range, from as low as few thousand bits per second to as high as several hundred megabits per second. Although in this mode the user will not wait for the response by sitting at a terminal (e.g., batch job submissions to supercomputers, transfers of non-real-time visualization data), bandwidth requirements will still be lower-bounded by the overall scheduling requirements of the study being completed using the PSE [Den96].

In general, in a good large-scale PSE system framework "bandwidth-greedy" material is distributed in a way that conserves bandwidth and allows support of a large number of simultaneous users. The distribution of tasks, across the network and across resources, will depend on the task complexity, desired schedules and resource constraints. The solutions should not rule out use of any network type (wire, optical, wireless) or access mode (high-speed and low-speed). For example, a possible distribution mix for a problem that requires use of parallel computing may include a user interface task on a portable computer in the field or classroom (perhaps using wireless to the closest high-speed network drop), a visualization, a computational model running on a remote personal workstation with data on a file server, and a communicating, larger, model running on remote supercomputer(s). To combat entropy, a distributed PSE will invariably have to centralize some of its functionalities, such as material updating, master backups and system evaluation. The exact mix and density of the "symbols," functions, and the content delivery modes that is most efficient remains a research issue. However, it is clear that a "quality" PSE needs to dynamically customize its CHI, and its presentation and communication modes, to match user expertise, user knowledge absorption rate, and the available computing and networking resources.

3.3.2 Delay and Jitter

End-to-end response delay can be a big problem in any PSE. EDSS and NovaNET related studies show that synchronous end-to-end interaction (round-trip) delays that consistently exceed about 250 ms are often unacceptable from the user point of view when the interaction is conducted in the key-stroke-by-keystroke mode [Bitz73, Kau95]. Furthermore, the video, voice and animation jitter should be less than about 10 ms in general, and for some specific coding approaches such as MPEG, less than 1 ms. Our measurements indicate that, except over limited areas, current incarnation of the Internet is probably not an adequate medium for key-by-key interactions. An alternative to real-time interaction on the key-by-key basis is for PSE to operate in semi-batch mode where the user interface and interactions are designed in such as way that a user expects some delays (not exceeding few tens of seconds), and does not consider long responses as system failures

For example, we have measured network delays on the North Carolina State University (NCSU) campus intranet, in the NC Research Triangle (about 40 miles per side) wide-area net, and over the Internet stretch between NCSU and University of Illinois at Urbana-Champagne (UIUC) [Kau95]. Table 1 illustrates the results. It shows the probability of response time for on-campus network and Internet under different loads. Assuming that a PSE application has response time of about 100 msec or better [Bal96], the network response times under 100 msec are considered *good*, response times between 100 and 150 milliseconds are considered *acceptable*, and response times over 150 msec are considered *poor*. The results show that a well designed campus network (or intranet) can adequately support modern PSEs, but problems grow rapidly beyond campus bounds. For instance, the NCSU-UIUC Internet link[8] was totally inadequate[9] for interactive PSE work during high traffic time slots (e.g., midday), and was at best marginal in medium to low traffic conditions [Kau95]. Adequate long-distance throughput over Internet is another problem.

Table 1: Campus and Internet response times under different traffic loads

Network	Traffic Load	Probability that Response Time is		
		Good	Acceptable	Poor*
NCSU Campus	Low	0.9963	0.0020	0.0017
	Medium	0.9889	0.0054	0.0057
	High	0.9566	0.0356	0.0078
Internet	Low	0.9682	0.0176	0.0142
	Medium	0.9502	0.0130	0.0368
	High	0.7187	0.0458	0.2355

(*) Includes lost packets.

Obviously, networks that provide delay, jitter and throughput guarantees, can make a big difference.

4. Summary and Conclusions

Traditionally QoS is associated with the performance of networks and network-based systems. However, the concept of **meta-(super)computing** [e.g., Bak96], in conjunction with the proliferation of a wide-variety of high-performance computing engines and networking technology, new network-based computing tools (e.g., PVM, MPI), and new standards (e.g., CORBA, TINA-C) acts a strong mixing factor between traditional quality characteristics of a PSE (e.g., correctness, reliability, ease of use) and its network component. While traditional PSE **architectures** tend to be centered around a single system, and possibly a single database on top of which a single workflow engine provides services to several clients, PSEs of the future will be much more distributed and diverse [e.g., Gal94], and issues such as *interoperability, concurrency, fault-tolerance, scalability, availability, interoperability, and general performance* become very important [Alo96].

This led us to broaden the classical definition of QoS to include not only network-based parameters (such as response delay and throughput), but also measurable end-user quality characteristics such as system availability, timing performance, algorithmic scalability, effectiveness, quality of system content, quality of user-system interactions, and so on. In order to facilitate integration of the QoS and PSE concepts, we view PSEs as computer and network-based systems that support **scientific workflows**, i.e., a series of structured activities and computations that arise in scientific problem-solving. This provides us with a natural mechanisms for formal specification and quality analysis of PSEs by using already existing techniques and approaches, such as queuing networks, Petri-nets, software reliability models, and so on.

Scientific workflows are expected to coexist, cooperate and meld with other user workflows (e.g., business workflows, educational workflows, legislative workflows) so the quality of these scientific workflows will have to be compatible. We have used information from existing systems and workflows to define some quantitative bounds on the quality of services that cooperating PSEs will have to meet . For example, we find that, from a usability perspective, the lower bound on

[8]Measurements were conducted in Spring and Summer 1995.

[9]Probability of good or acceptable response was about 0.76, i.e., far less than 0.86 that NovaNET users consider the lower bound.

guaranteed system availability should be 0.95 or better, while acceptable synchronous user-machine interactions require user-level round-trip response times that are consistently less than about 250 msec, and network-level round-trip delay times less than about 150 msec. Use of multimedia imposes additional restrictions on parameters such as jitter and data loss probability, while end-user risks impose bounds on the accuracy and reliability of numerical computations and algorithms.

It is our belief that the next generation of PSEs must have QoS parameters designed into the system, rather than coerced into the system through operational feedback, as is the case with many existing PSEs, or these new PSEs will fail to live up to both user needs and user expectations.

5. References

[Alo96] Gustavo Alonso and Hans-Jorg Schek, "Research Issues in Large Workflow Management Systems", Proc. NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-art and Future Directions, (http://optimus.cs.uga.edu:5080/activities/NSF-workflow/proc_cover.html), Edited by Amith Sheth, U. of Georgia, Athens, GA, May 8-10, 1996

[Amb95] J. Ambrosiano, R.Balay, C. Coats, A. Eyth, S. Fine, D. Hils, T. Smith, S. Thorpe, T. Turner, and M. Vouk, "The Environmental Decision Support System: Air Quality Modeling and Beyond", Proceedings of the U.S. EPA Next Generation Environmental Modeling Computational Methods (NGEMCOM) Workshop, Bay City, Michigan, August 7-9, 1995 .

[Bak96] Mark Baker and Geoffrey Fox, "Metacomputing: The Informal Supercomputer" Proc. NSF Train the Trainer Workshop, Cornell Theory Center, Ithaca, NY, May 6-9, 1996 (http://renoir.csc.ncsu.edu/RTCPP/HTML/Workshop2/index.html)

[Bal96] R. Balay, M. A. Vouk, and H. Perros "A Lightweight Software Bus for Prototyping Problem Solving Environments", Accepted for the Special Session on Networks and Distributed Systems in the Eleventh International Conference on Systems Engineering, Las Vegas, 1996.

[Beg93] Adam Beguelin, J. Dongarra, Al Geist, Robert Manchek, K. Moore, and Vaidy Sunderam, "PVM and HeNCE: Tools for Heterogeneous Network Computing," Environments and Tools for Parallel Scientific Computing, Edited by Jack Dongarra and Bernard Tourancheau, Advances in Parallel Computing, Volume 6, North-Holland, 1993.

[Beg94] A. Beguelin, J. Dongarra, A. Geist, and R. Manchek, "HeNCE: A Heterogeneous Network Computing Environment , '' Scientific Programming, Vol. 3, No. 1, pp 49--60., 1994

[Bel89] BELLCORE, *Network Switching Element Outage Performance Monitoring Procedures*, SR-TSY-000963, Issue 1, April 1989.

[Bel90a] BELLCORE, *The Analysis and Use of Software Reliability and Quality Data*, SR-TSY-001547, Issue 1, January 1990.

[Bel90b] BELLCORE, *Reliability and Quality Measurements for Telecommunications Systems (RQMS)* , TR-TSY-000929, Issue 1, June 1990.

[Bit73] M. D. Bitzer and D.L. Bitzer, "Teaching nursing by computer: an evaluation study," Computers in Biology and Medicine, Vol 3 (3), pp. 187-204, 1973.

[Boe89] Boehm, B.W., *Tutorial: Software Risk Management*, IEEE CS Press, 1989.

[Coa96] Coats, C, *private commun ication*, 1996.

[Cra92] Cramp R., Vouk M.A., and Jones W., "On Operational Availability of A Large Software-Based Telecommunications System," Proceedings Third International Symposium on Software Reliability Engineering, IEEE CS, 1992, pp. 358-366.

[Den96] R.L. Dennis, D.W. Byun, J.H. Novak, K.J. Galluppi, C.C. Coats, M.A. Vouk, "The Next Generation of Integrated Air Quality Modeling: EPA's Models-3," Atmospheric Environment, Vol 30 (12), pp 1925-1938, 1996.

[Dik96] P. Dikshit and M.A. Vouk, "Reliability and Availability of a Wide-Area Education System," submitted to ISSRE96, April 1996.

[EBV95] Elmaghraby S.E., Baxter E.I., and Vouk M.A., "An Approach to the Modeling and Analysis of Software Production Processes," Intl. Trans. Operational Res., Vol. 2(1), pp. 117-135, 1995.

[Ell79] C. A. Ellis, "Information Control Nets: A Mathematical Model of Office Information Flow", Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems, 1979.

[Elm66] Elmaghraby S.E., "On generalized activity networks," J. Ind. Eng., Vol. 17, 621-631, 1966.

[Elm92] A.K. Elmagarmid, "Database Transaction Models for Advanced Applications", Morgan Kaufmann, 1992.

[Gal94] S. Gallopoulos, E. Houstis and J.R. Rice, "Computer as Thinker/Doer: Problem-solving environments for computational science, IEEE Computational Science and Engineering, Vol. 1, pages 11-23, 1994.

[Gei94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck, and V. Sunderam, "PVM - Parallel Virtual Machine - A User's Guide and Tuotrial for Networked Parallel Computing," MIT Press, 1994.

[Geo95] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Databases, Vol. 3(2), April 1995.

[Gra93] James Gray and Andreas Reuter, "Transaction Processing: Concepts and Techniques," Morgan Kaufmann, 1993.

[Hal77] Maurice H. Halstead, *Elements of Software Science*, Elsevier, 1977.

[How80] W.E. Howden, "Applicability of software validation techniques to scientific programs," ACM Transactions on Programming languages and Systems, Vol 2 (3), 1980.

[Jon96] Jones W. and Vouk M.A., "Software Reliability Field Data Analysis," Chapter 11 in *Handbook of Software Reliability Engineering*, editor M. Lyu, McGraw Hill, pp. 439-489, January 1996.

[Kau95] P.K. Kauer, *An Analysis of North Carolina State University's Network Performance*, M.S. Thesis, Department of Computer Science, NCSU, 1995.

[Ley93] F. Leymann and W. Altenhuber, "Managing Business Processes as an Information Resource", IBM Systems Journal, Vol. 33(2), pp. 326-348, 1994. 13. M. Hsu (ed.), "Special Issue on Workflow and Extended Transaction Systems", IEEE Data Engineering, Vol. 16(2), June 1993.

[Lyu95] *Software Fault-Tolerance* editor M. Lyu, John Wiley & Sons, 1995.

[Lyu96] *Handbook of Software Reliability Engineering*, editor M. Lyu, McGraw Hill, 1996.

[McA96] McAllister D.F. and Vouk M.A., "Software Fault-Tolerance Engineering," Chapter 14 in *Handbook of Software Reliability Engineering*, editor M. Lyu, McGraw Hill, pp. , January 1996.

[Nov96] NovaNET on-line reliability estimation interface, May 1996.

[Qua54] Henry Quastler, Editor, *Information Theory in Psychology: Problems and Methods*, Proc. of a Conf. on the Estimation of Information Flow, Monticello, IL, July 1954, The Free Press, Glencoe, IL, 1955.

[Rin95] A. Rindos, M. Vouk, S. Woolet, J. Hines, and J. Lester, "ATM Technology Enabling Educational Applications Across the North Carolina Information Highway", in Proc. TELECOM '95 FORUM Technology Summit, ITU, Geneva, pp. 519-522, October 1995.

[Sin94] M.P. Singh and M.N. Huhns, "Automating Workflows for Service Provisioning: Integrating AI and Database Technologies," IEEE Expert, Vol. 9(1), October 1994.

[Sin95] M.P. Singh, "Semantical Considerations on Workflows: Algebraically Specifying and Scheduling Intertask Dependencies," Proceedings of the 5th International Workshop on Database Programming Languages (DBPL), September 1995.

[Sin96a] M.P. Singh, "Synthesizing Distributed Constrained Events from Transactional Workflow Specifications," Proceedings of the 12th International Conference on Data Engineering (ICDE), March 1996.

[Sin96b] M.P Singh and M.A. Vouk "Scientific Workflows: Scientific Computing Meets Transactional Workflows," Proc. NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-art and Future Directions, http://optimus.cs.uga.edu:5080/activities/NSF-workflow/proc_cover.html, Edited by Amith Sheth, U. of Georgia, Athens, GA, May 8-10, 1996

[Sni95] M. Snir, S.W. Otto, S. Huss-Lederman, D.W. Walker, and J. Dongarra, "MPI - The Complete Reference, " MIT PRess, 1995.

[Str67] John M. Stroud, "The Fine Structure of Psychological Time," Annals of the New York Academy of Sciences, Vol 138, Art. 2, pp 623-631, February, 1967.

[Wac92] Helmut Wachter, Andreas Reuter, “Contracts” In [Elm92].

[Wai96] Jacques Wainer, Mathias Weske, Gottfried Vossen, Claudia M Bauzer Medeiros, "Scientific Workflow Systems," Proc. NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-art and Future Directions, http://optimus.cs.uga.edu:5080/activities/NSF-workflow/proc_cover.html, Edited by Amith Sheth, U. of Georgia, Athens, GA, May 8-10, 1996