SAND99-2641J

Computational Environment and Software Configuration Management of the 1996 Performance Assessment for the Waste Isolation Pilot Plant

G. K. Froehlich[1], C. M. Williamson[2], and H. C. Ogden[3]

[1]*Regulatory Compliance Department, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185, USA*
[2]*Networks and Systems Integration Services, Compaq Computer Corporation*[*]*, 5951 Jefferson NE, Albuquerque, NM 87109, USA*
[3]*Technical Integration Department, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185, USA*

Abstract

The US Department of Energy (DOE) Waste Isolation Pilot Plant (WIPP), located in southeast New Mexico, is a deep geologic repository for the permanent disposal of transuranic waste generated by DOE defense-related activities. Sandia National Laboratories (SNL), in its role as scientific advisor to the DOE, is responsible for evaluating the long-term performance of the WIPP. This risk-based Performance Assessment (PA) is accomplished in part through the use of numerous scientific modeling codes, which rely for some of their inputs on data gathered during characterization of the site. The PA is subject to formal requirements set forth in federal regulations. In particular, the components of the calculation fall under the configuration management and software quality assurance aegis of the American Society of Mechanical Engineers (ASME) Nuclear Quality Assurance (NQA) requirements. This paper describes SNL's implementation of the NQA requirements regarding configuration management. The complexity of the PA calculation is described, and the rationale for developing a flexible, robust run-control process is discussed. The run-control implementation is described, and its integration with the configuration-management system is then explained, to show how a calculation requiring 37,000 CPU-hours, and involving 225,000 output files totaling 95 Gigabytes, was accomplished in 5 months by 2 individuals, with full traceability and reproducibility.

Keywords: Configuration management; Run control; Run management; Traceability; Reproducibility; Quality assurance; Performance assessment; Waste Isolation Pilot Plant; Transuranic waste; Radioactive waste

[*] At the time the work was performed, this was Digital Equipment Corporation (DEC)

1. Introduction

The Waste Isolation Pilot Plant (WIPP) is a deep geologic repository located in southeast New Mexico, which has been licensed for the permanent disposal of transuranic waste generated by US Department of Energy (DOE) defense-related activities [1]. The scientific advisor to DOE, Sandia National Laboratories (SNL), is responsible for evaluating the long-term (10,000-year) performance of the WIPP. This risk-based Performance Assessment (PA) is accomplished in part through the use of numerous scientific modeling codes, which rely for some of their inputs on data gathered during characterization of the site. Due to the regulatory nature of any nuclear waste disposal project, the participants must be held to high standards of accountability. For SNL, this means that the performance assessment results are of sufficient import, and complexity, as to require rigorous accountability and control of all versions of the inputs, codes, outputs, and the multiple relationships between and among them. This accountability and control is accomplished through configuration management and run control.

The origins of the formal software configuration management (SCM) requirements are given, and the status of SCM at the time the requirements were imposed is discussed. This is followed by a detailed description of SNL's implementation of SCM, including a description of procedures and roles. Actual examples are cited which demonstrate the effectiveness of the SCM system. Finally, there is a discussion of important lessons learned.

Following a description of the computational complexity involved, a case is made for rigorous run-control and job-distribution processes. Again, the initial status of run control at SNL is discussed, followed by a discussion of our implementation of an improved, more robust and effective process. Once again, examples of the system's effectiveness are cited, followed by a discussion of lessons learned.

2. Software Configuration Management

Software configuration management (SCM), as implemented for software on the WIPP project, is comprised of both configuration management and change control. For our purposes, configuration management can be defined as the meticulous identification, storage, and ongoing tracking of computer codes, from a baseline version through all subsequent versions, along with all relevant inputs, outputs, compilation options, library linkages, and any other information needed to faithfully reproduce the most recent or any previous calculation for which a code has been used (whether for testing or for production). Change control is the formal process by which proposed code changes are evaluated for importance and impact, and only formally approved changes are allowed. All changes are tracked and all versions are maintained by the configuration management system.

Origin of SCM Requirements

In 1992, the Land Withdrawal Act [2] named the US Environmental Protection Agency (EPA) as the regulator for WIPP. As such, EPA became responsible for developing disposal regulations, and for certifying the long-term safety of the repository. In late 1993, Regulation 40CFR191 [3] set forth the disposal regulations and release limits. In effect, this regulation outlined *what* needed to be done to demonstrate compliance with the release limits, without specifying *how* to do it. Then, in early 1996, 40CFR194 [4] established criteria for demonstrating compliance with 40CFR191, in effect specifying *how* to do so. This latter regulation invoked the American Society of Mechanical Engineers' (ASME) Nuclear Quality Assurance (NQA) Standards, *i.e.*, ASME NQA-1-1989 edition, NQA-2a-1990 addenda (Part 2.7) to ASME NQA-2-1989 edition, and ASME NQA-3-1989 edition [5]. As an integral part of its software QA requirements, the NQA standards require configuration management and change control. These requirements, coupled with the requirement to support SNL's software QA program through version control and documentation of software testing [6], combined to dictate the SCM system design. The design was further influenced by the complexity of the PA calculations. This complexity was a result of the physical processes being modeled, as well as the need to model uncertainty through the use of probabilistic methods.

An example of this complexity is illustrated by Figures 1 and 2. Figure 1 shows, at a high level, the major processes modeled by the WIPP PA codes, as well as their relationships to one another. Each of these processes consists of the modeling code, as well as numerous lower-level support codes required to implement the model. Thus, for example, the "Regional and Local Groundwater Flow" model shown in Figure 1 consists of the modeling code and support codes shown in Figure 2. A similar expansion exists for every process shown in Figure 1. In addition, this complex suite of codes must be exercised a large number of times to properly account for statistical uncertainty [7, 8].

Establishment of Existing SCM Condition

The interactions between EPA and SNL began in earnest in early 1994. At that time, SNL embarked on a process of explaining to the EPA the various PA modeling codes, and their linkages to one another and to the site-characterization data. The results of a preliminary PA, conducted in 1992 [9], were to be used as the basis for these interactions. It quickly became apparent that greater traceability and reproducibility were needed. The burden of file management and version tracking had been placed upon the individual code developers. No formal, centralized configuration-management procedures existed, nor was any mechanism in place to facilitate the process. Consequently, some of the practices were closer to the realm of "tribal knowledge" than to modern information management. Formal reviews of the program, conducted by DOE, served to emphasize further the need for rigorous control and tracking of changes to, and versions of, computer codes and their inputs and outputs.

Changing ingrained practices requires changing an organization's culture, which cannot be accomplished without strong management support. Further, the suggested solution must show a clear benefit to those being asked to apply it. This was the challenge facing WIPP PA.

Implementation of Software Configuration Management

Implementation of SCM involved several tasks. The first task was to determine what types of SCM tools were available, and to select one that met the project's needs. Simultaneously, these needs had to be codified and developed into project procedures, and access-control requirements had to be identified. The next task was to identify all codes that had been used, or might be used, in the PA calculation to support the Compliance Certification Application (CCA) and submitted to EPA [10], and then to place those codes under SCM. A "primitive baseline inventory list" was created, indicating the version, state (*e.g.*, in test or in production), and status (*e.g.*, under revision or not under revision) for all versions of all codes.

Virtually all of the WIPP codes had been developed under the OpenVMS operating system, sold by Digital Equipment Corporation (DEC). Establishing SCM on the same platform avoided several problems with traceability and compatibility that file transfer between different platforms would have caused. Another DEC product, DECSet Tools, was selected after it was evaluated and determined suitable. A consultant with configuration management experience was provided by DEC Networks and Systems Integration Services to design and implement a system that would meet our requirements, using the DECSet Tools product. The components of this product which were used included source-code control, dynamic and static code analysis, and build-control modules.

During the process of bringing all codes under control of the SCM system, it became clear that most of the codes adhered to some individual naming conventions, but did not follow a more global naming convention that allowed numerous codes to be managed at the same time. Thus, a code naming convention was adopted that used a unique "code prefix" for each code and every file (source and otherwise) associated with the code. In addition to a naming convention, several consistency issues were also addressed (*e.g.*, calling external files using logical, rather than literal, references; removing "hard-wired" constants from codes, and instead, retrieving them from a shared database). These tasks were accomplished by requiring code developers to complete a single-page checklist, covering the above issues, prior to placing codes under SCM for the first time. This facilitated the identification of code elements (*e.g.*, source, libraries, other modules), as well as the creation and subsequent use of the appropriate working environments.

Two primary roles, SCM Coordinator and SCM Librarian, were identified as necessary for successfully implementing and managing the SCM process, and so the next task was to select qualified individuals to fill these roles. The SCM Coordinator was responsible for maintaining change control records for all codes, as well as maintaining the Software Baseline Inventory List, which is a listing of software that has been approved for use, retired from use, is a candidate for retirement, or is in the process of qualification. The SCM Librarian was responsible for the design, implementation, and day-to-day management of the SCM system. DEC's Networks and Systems Integration Services filled this latter role.

The final task in the SCM implementation process was to design and implement the working environments that were needed to support different activities, and different security requirements, for codes in different phases of development. To provide appropriate security, each code was placed in its own SCM storage area, and assigned its own development, test, and production environments. Each developer was given the option of maintaining his/her own development environment for day-to-day activities. Each code was given a test environment, where development and integration testing would take place. Once the SCM Coordinator determined that a code was ready for a production build (*i.e.*, compilation and linking), the build was done in a production environment by the SCM Librarian, and then subjected to acceptance testing in the test environment. Production builds were scripted and executed directly within the SCM system. Build logs and the resulting production executable were stored in the SCM system after build completion. Only the SCM Librarian was allowed write-level access to the production environment, *i.e.*, no one else could alter the contents. After completion of acceptance testing, the code was released for general use and made available in the production environment. All subsequent official calculations were conducted in the production environment.

Two additional computational environments were created, each nearly identical to the one created for the CCA calculation. One of these was for the EPA, so that they could conduct calculations of their own, using our codes as well as some of their own, with the same benefits of traceability and reproducibility [11]. The second such environment was created for oversight groups and other interested stakeholders. These parties were responsible for their own SCM administration, and SNL did not have access to their files, except that we all shared the same system administrators. EPA and interested stakeholders, of course, did have full read-level access to the SNL production environment.

Effectiveness

Internal project reviews, DOE surveillances, and audits were conducted at various times before, during, and following the CCA calculation and submittal. The large number of runs and the very large number of files generated interest in our ability to retrieve and reproduce results. Early in the review process, DOE found our implementation of SCM to be commendable, and cited it as a "best practice". Complete retrievability and reproducibility were demonstrated for all SCM-controlled files, from any and all phases of the calculation. In addition to the "auditability" aspects of the system, analysts and other consumers of the files could be confident that the SCM components they needed for their work (*e.g.*, executables, inputs, calculation results) were accessible, authentic, and always in the right place.

The formal PA calculation for the CCA involved 37,000 CPU-hours (over 4.2 CPU-years), with over 225,000 output files retained. Many times that number of files (on the order of 2 million) were created and retained temporarily as intermediate results. The files relegated to permanent storage occupy some 95 Gigabytes, out of a total of over three Terabytes generated. Due to the rigorous naming convention, every single file can be readily identified, as can its role in the overall calculation. For example, each file name contains a reference to the relevant codes, the particular calculation (*e.g.*, CCA), and such calculation-specific details as vector number, scenario number, and statistical replicate. Any of the temporary, intermediate files can be regenerated on demand. Many times, we have been called upon to locate and retrieve a specific file or set of files, and each time we have been able to respond quickly and accurately. Significantly, the SCM Librarians have been the individuals responding to such requests, thus removing this burden from the analysts and code sponsors.

What Worked Best

Administration of the SCM system was limited to a small number (two) of individuals, the SCM Librarians. This in itself brought a certain level of consistency to the process. In addition, it permitted the SCM resources (*e.g.*, disk space, code libraries) to be controlled and managed more easily, since the managers could see the "big picture". A bonus was the coordination effect that the SCM Librarians provided between teams.

Not surprisingly, many code sponsors and their teams initially resisted the need for project-wide file-naming conventions. Their concerns generally related to perceptions of "ownership". The very aggressive schedule only added to their anxiety. However, the naming (and other) conventions provided the additional consistency that was needed to permit the ready identification and retrieval of a specific file out of the thousands stored under SCM. Once a user (analyst, manager, regulator, stakeholder) was introduced to the environment, they were able to easily find any input, code, or result they needed, along with a complete history of all versions. The facility with which the system kept track of versions, changes, inputs, outputs, etc., soon led code developers to rely on the SCM system even for their day-to-day code-development work.

Lessons Learned

Two important lessons surfaced during the implementation and subsequent use of the WIPP PA software configuration management system. The first is that SCM (including change control) should be applied early in the life cycle of any software product. Trying to establish a history after the fact is usually very difficult, almost always inaccurate, and expensive. And having implemented SCM, its application must be ongoing and relentless. Any lapses lead to loss of control, and it requires much more work to regain control than it does to maintain it. The temptation to skip the formalities due to schedule (or other) pressures must be resisted. This, again, requires strong management support.

3. Run Management and the Computational Environment

For SNL's WIPP PA process, run management or run control can be defined as the automated execution of a suite of codes by those granted the necessary access, including retrieval of all needed codes and inputs from within software configuration management, as well as the appropriate disposition of outputs. Further, run control addresses the distribution of the computational load across appropriate and/or available resources.

Requirements for the Computational Environment

No formal requirement dictated the use of a particular run-control or job-distribution methodology. Prudence, however, suggested that some kind of automated run-control scheme was appropriate. The sheer complexity of the simulations, the vast numbers of files involved, and the linkages from inputs to codes, between codes, and from codes to outputs required great care to avoid mistakes. This, coupled with the need for full traceability, reproducibility, and retrievability, lobbied against manual run control. Further, run scripts (*i.e.*, lists of directives which automate the execution and distribution of the runs) could be qualified once, then used many times, whereas manual runs would each be subject to individual qualification. In addition, once in place, a properly implemented scheme would permit better distribution of runs across available resources, enabling us to meet our very aggressive schedule. Another benefit of such a system is the facilitation of more accurate scheduling and of better forecasting (of run-completion times). The automated disposition of results into the SCM system would also serve to expedite the availability of results for further analysis or review. In fact, the existence of the SCM system greatly facilitated the automation of run control.

Establishment of Existing Computational Environment

Prior to implementation of the formal SCM system, calculations were constructed and submitted almost entirely manually. There was software intended to act as a "run executor", but construction of the runstreams was a manual process, as were job submittal and disposition of outputs. Each code sponsor was responsible for the submission, tracking, and disposition of his/her own jobs. Often there was insufficient communication between code sponsors, especially for codes that were near opposite ends of a runstream. Contention for resources was high, disk space was inefficiently utilized, and as previously mentioned, any naming conventions used were local rather than global.

Implementation of the Computational Environment

As already mentioned, the WIPP codes had been developed under the OpenVMS operating system. OpenVMS is a multi-user, interactive and batch operating system that provides time-sharing between all users. To prepare for the CCA PA calculation, a distributed calculation environment was created, utilizing 13 OpenVMS systems, including three single-processor machines and 10 four-processor machines (for a total of 43 available processors). Eight of these processors were generally reserved for development and system-overhead tasks, leaving 35 processors available for calculations. The 13 systems were configured as a Network Interface (NI) cluster. NI clusters are loosely coupled collections of systems that have the ability to share user information and peripherals (such as disk drives). A total of 400 Gb of disk space was configured to be available to all nodes in the cluster. The disk drives were mostly configured as RAID (Redundant Array of Independent Disks) to maximize throughput and minimize downtime related to disk failures. Each system was assigned a calculation disk that was directly attached, to improve throughput (input/output performance). This calculation disk provided a working area for jobs running on the local CPU(s).

Given the massive number of individual job runs required to complete the calculation, a mechanism was needed to provide for automated job submission and tracking. Batch execution queues were established on each system. A generic queue (*i.e.*, one not associated with a particular computational node) was created that directed pending jobs to the next available system queue as processors became available.

All official runs were controlled by two Run Coordinators, known as the "CCA Masters". These individuals made all the decisions about utilization of CPU time and other system resources for the CCA. A major part of the Run Coordinators' jobs was to balance resource utilization between automated runs and ongoing analysis tasks. Throughout the calculation, analysts reviewed calculation results as they became available. The need to provide system resources for these sometimes CPU-intensive analysts' functions varied on a day-to-day basis, and had an impact on the number of processors available for calculations.

The complexity of the CCA calculation required a level of run management or run control beyond what would normally be expected of commercial job scheduling packages. The calculation required about 59,000 batch jobs, each involving from 2 to 30 different executables, and required dynamic input identification and output naming.

A run-control system was developed by scripting all code flows (runs), setting up a job distribution queuing system, capitalizing on the file naming convention, and interacting directly with the SCM system from the scripts. In most cases three types of scripts were developed — distribution control scripts to submit and distribute large numbers of runs, preliminary job scripts to run common precursor jobs, and calculation run scripts to run the many thousands of individual jobs. Run distribution scripts received inputs that indicated the number of jobs being run, as well as flags instructing the distribution script to run preliminary or calculation jobs. All scripts reported any errors or problems directly to the Run Coordinators via email, allowing them to easily audit many runs and give their direct attention to any problems.

For example, for one of the major subsystems the job execution flow is as follows:

- Execute the subsystem distribution script with inputs that will cause any preliminary jobs to be run in batch mode. All necessary outputs from the preliminary jobs are automatically placed under SCM and identified as relating to the current calculation.

- Once the preliminary job(s) are completed, the Run Coordinator executes the subsystem distribution script, using input flags that cause the distribution script to submit many individual runs of the calculation to batch queues. For the WIPP CCA calculation, anywhere from 100 to 5000 jobs were generated at a time for this particular subsystem.

Each script execution, whether for preliminary or for calculation jobs, worked directly with SCM to get copies of relevant executables and input files, and to place important output files under SCM, using knowledge of the calculation naming convention.

Effectiveness

The formal PA calculation for the CCA involved on the order of 50 codes altogether, represented by some 21 code sponsors. The calculation took five months end-to-end (March through July, 1996), and was conducted by only two people (the Run Coordinators). As previously stated, the CCA calculation involved 37,000 CPU-hours (over 4.2 CPU-years), with over 225,000 files retained. The 59,000 batch jobs, each comprised of runstreams invoking from 2 to 30 executables, translate to over 700,000 individual code executions.

Any of the temporary, intermediate files can be regenerated on demand. Many times SNL has been called upon to reproduce a result, and each time we have been able to respond quickly and accurately. In such cases, the Run Coordinators have been the individuals responding to the requests, thus removing this burden from the code sponsors. The latter are, of course, still responsible for defending their codes, their choice of inputs, and so on.

Once the codes had been scripted and benchmarked, reliable run-completion schedules became straightforward to produce. The existence of run-control scripts also provided reusability, since the scripts could be invoked as often as needed. SNL has, in fact, had several opportunities to reuse the system. One example was an exercise conducted for the EPA, which consisted of changing many of the input parameters and conducting the entire calculation again, in order to evaluate the impact of those changes [12]. This exercise was conducted by the Run Coordinators, and took only three months. The magnitude of the calculation was about the same as for the CCA.

What Worked Best

A concern at the beginning of the calculation was how the resources could be managed so that the massive calculation could be completed on schedule, while a diverse group of regulators, analysts, and other users conducted their work. Since management of the calculation had been limited to only two Run Coordinators, the computational resources were easily controlled and managed. Once again, due to the small number of individuals in control of the overall process, coordination between teams was facilitated. Since the Run Coordinators controlled all the computational resources and had responsibility for all the runs, they were the logical point of contact for analysts wanting to know when their data would be ready, for providing management with tracking schedules, for anyone wanting to know what was in process, for anyone requesting additional computational resources, and so on.

Again, many code sponsors and their teams initially resisted the concept of Run Coordinators to run "their" codes, and again the very aggressive schedule added to their anxiety. However, the scripting of runs and distribution of jobs resolved many of the conflicts that typically resulted from doing business the old way. Users were isolated from system-resource issues, and could be given reliable estimates of job-completion dates. The tight coupling with SCM meant that the status of jobs and the whereabouts of job components could be tracked at any point before, during, and after a calculation.

Lessons Learned

Complex, large-scale calculations require an integrated computational environment to effectively maintain traceability and reproducibility. Further, in a regulatory environment, where rigorous audits are the rule, there is great advantage in removing as many manual steps as possible. While creation of automated scripting and file-management software requires much up-front effort and takes a lot of resources, such software can be tested and qualified once, and then used repeatedly. The manual alternative requires qualification of the manual steps each time they are performed. For large numbers of calculations, or for ones involving a large number of steps, this repeated manual qualification always ends up taking more time and resources than are required for the automation of the processes. Do not try to do this any other way!

Once again, there was enthusiastic buy-in by code users once the benefits of the run-control environment became apparent. No longer did they need to concern themselves with finding available resources, probing the system for information about job status, or resolving schedule conflicts. In fact, just as with the SCM system, the run-control environment proved so appealing that code sponsors now use it for their day-to-day activities.

Lastly, the importance of the reusability of the automated run-control environment cannot be over-emphasized. Never assume that a calculation will be done only once; it is axiomatic among PA professionals that any large analysis will be redone at least once. For every WIPP calculation SNL has performed, we have been required (*e.g.*, by the regulator) to perform the calculation over, with minor changes to selected inputs. To date, we have done formal calculations, of approximately the same magnitude as the CCA calculation, some five additional times.

4. Conclusions

For complex, large-scale calculations, conducted in a regulatory environment, all components must be carefully managed. Run control and configuration management are paramount for providing traceability and reproducibility. Even in a non-regulatory environment, they are good practice. Elimination of manual steps, to the extent practicable, further enhances credibility and accountability. The introduction of configuration management and run control into an environment in which they have not previously been implemented or rigorously applied inevitably causes concern and suspicion among the users. To allay these fears, the benefits of the proposed environments must be demonstrated to users. For the benefits (to the users and to the project) to be fully realized, change control, configuration management, and run control must be applied without any exceptions. This requires strong management support. On the WIPP project, once these systems were in place and users had gained experience with them, the users became enthusiastic supporters.

Additional benefits to the project included more efficient use of resources and the ability to schedule calculations more accurately. Efficient use of resources was not limited to computational hardware, either. For example, code sponsors and analysts were freed from responding to the very frequent requests by regulators and stakeholders to provide results, codes, inputs, or even to reproduce a calculation, as this could be done by the SCM Librarians and/or Run Coordinators.

Acknowledgments

Work performed for Sandia National Laboratories (SNL), which is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. Review provided at SNL by J. Helton, A. Treadway, and A. Kane.

References

[1] Rechard, R.P., Historical Background on Performance Assessment for the Waste Isolation Pilot Plant, *Reliability Engineering and System Safety* (in this issue).

[2] Waste Isolation Pilot Plant Land Withdrawal Act, Public Law 102-579, (106 Stat. 4777), 1992.

[3] U.S. Environmental Protection Agency, 40 CFR Part 191: Environmental Radiation Protection Standards for the Management and Disposal of Spent Nuclear Fuel, High-Level and Transuranic Radioactive Wastes; Final Rule, *Federal Register*, 1993, **58**(242), 66398-66416.

[4] U.S. Environmental Protection Agency, 40 CFR Part 194: Criteria for the Certification and Re-Certification of the Waste Isolation Pilot Plant's Compliance With the 40 CFR Part 191 Disposal Regulations; Final Rule, *Federal Register*, 1996, **61**(28), 5224-5245.

[5] American Society of Mechanical Engineers (ASME) Nuclear Quality Assurance (NQA) Standards, ASME NQA-1-1989 edition, *Quality Assurance Program Requirements for Nuclear Facilities*; ASME NQA-2a-1990 addenda, part 2.7, to ASME NQA-2-1989 edition, *Quality Assurance Requirements for Nuclear Facility Applications*; and ASME NQA-3-1989 edition, *Quality Assurance Program Requirements for the Collection of Scientific and Technical Information for Site Characterization of High-Level Nuclear Waste Repositories*; American Society of Mechanical Engineers, Fairfield, NJ, 1989-1992.

[6] Froehlich, G.K., Ogden, H.C., & Byle, K.A., Software Quality Assurance in the 1996 Performance Assessment for the Waste Isolation Pilot Plant, *Reliability Engineering and System Safety* (in this issue).

[7] Helton, J.C., Davis, F.J., & Johnson, J.D., Characterization of Stochastic Uncertainty in the 1996 Performance Assessment for the Waste Isolation Pilot Plant, *Reliability Engineering and System Safety* (in this issue).

[8] Helton, J.C., Martell, M.-A., & Tierney, M.S., Characterization of Subjective Uncertainty in the 1996 Performance Assessment for the Waste Isolation Pilot Plant, *Reliability Engineering and System Safety* (in this issue).

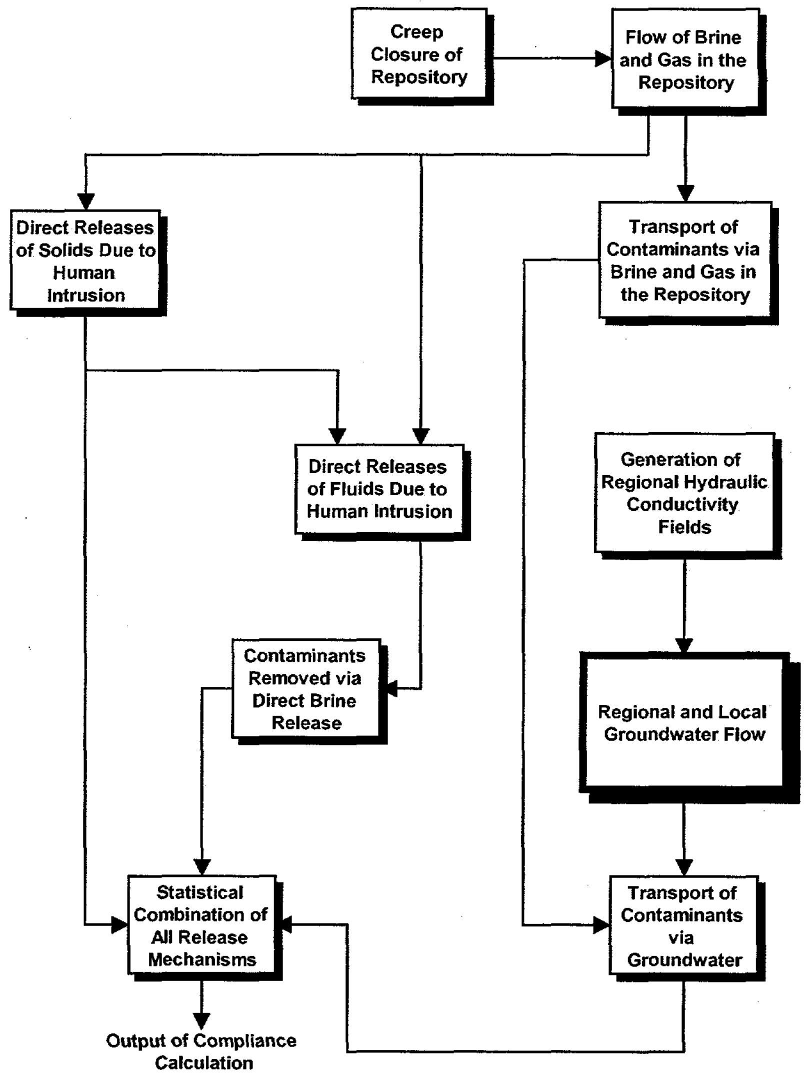
[9] WIPP PA (Performance Assessment) Department, *Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992*, Volumes 1-5, SAND92-0700/1-5, Sandia National Laboratories, Albuquerque, NM, 1992-1993.

[10] U.S. Department of Energy, *Title 40 CFR Part 191 Compliance Certification Application for the Waste Isolation Pilot Plant*, DOE/CAO-1996-2184, Volumes I-XXI, U.S. Department of Energy, Waste Isolation Pilot Plant, Carlsbad Area Office, Carlsbad, NM, 1996.

[11] Froehlich, G.K., Schneider, J.T., & Williamson, C.M., *SNL Participation in PA-Parameter Sensitivity Analysis for EPA*, Sandia WIPP Central Files, WPO#46853, Sandia National Laboratories, Albuquerque, NM, August 1997.

[12] Aragon, K.M., Williamson, C.M., Froehlich, G.K., & Schneider, J.T., *SNL Fulfillment of EPA-Mandated Performance Assessment Verification Calculation*, SWCF-A:1.2.07.4.1:PA:QA, Sandia WIPP Central Files, WPO#46854, Sandia National Laboratories, Albuquerque, NM, August 1997.

List of Figures



Creep Closure of Repository
Flow of Brine and Gas in the Repository
Direct Releases of Solids Due to Human Intrusion
Transport of Contaminants via Brine and Gas in the Repository
Direct Releases of Fluids Due to Human Intrusion
Generation of Regional Hydraulic Conductivity Fields
Contaminants Removed via Direct Brine Release
Regional and Local Groundwater Flow
Statistical Combination of All Release Mechanisms
Transport of Contaminants via Groundwater
Output of Compliance Calculation

Fig. 1. Processes modeled by WIPP PA codes.

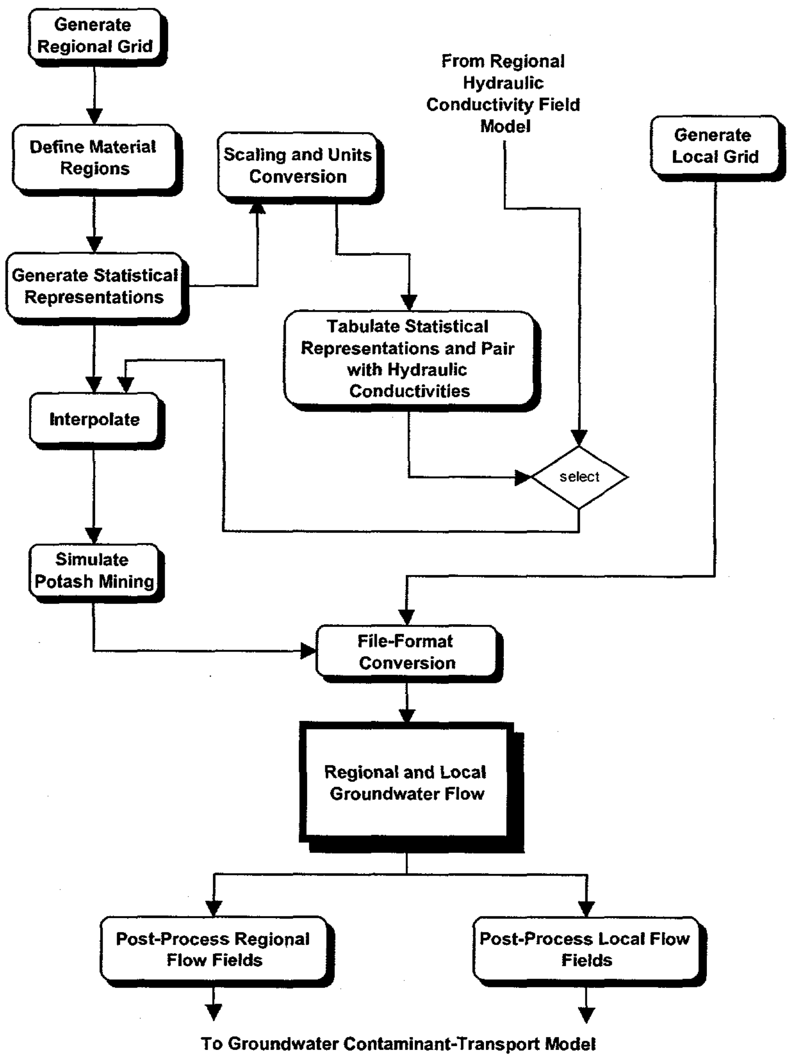

Generate Regional Grid
Define Material Regions
Scaling and Units Conversion
From Regional Hydraulic Conductivity Field Model
Generate Local Grid
Generate Statistical Representations
Tabulate Statistical Representations and Pair with Hydraulic Conductivities
Interpolate
select
Simulate Potash Mining
File-Format Conversion
Regional and Local Groundwater Flow
Post-Process Regional Flow Fields
Post-Process Local Flow Fields
To Groundwater Contaminant-Transport Model

Fig. 2. Expansion of "Regional and Local Groundwater Flow" model from Fig. 1, to include support codes.