*Proceedings of the 1990 Winter Simulation Conference*
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

# SYSTEM PERFORMANCE ANALYSIS WITH AN ADA PROCESS MODEL DEVELOPMENT

Joseph A. Viceroy

TRW Systems Integration Group
Redondo Beach, California

## ABSTRACT

Most ADPE performance models are used to support systems engineering analysis in proposals or the early phases of projects. These models verify performance requirements at a high level to validate system designs. Use of some of these models has continued through the life of a project analyzing the performance of the system as the design evolves.

The Command Center Processing and Display System-Replacement (CCPDS-R) Performance Model has enjoyed an expanded role on the CCPDS-R project. The performance model has not only influenced the system design, but system allocations of resources as well. CCPDS-R performance analysis and requirements compliance insight have been substantially enhanced over the early phase of development. This paper will reflect on the benefits of this approach drawing on real world experience gained to date on CCPDS-R.

## 1. PROJECT BACKGROUND

The Command Center Processing and Display System Replacement (CCPDS-R) program is a command and control system that provides missile warning information to various end users. The software for this project is being developed in Ada using a demonstration based, incremental development approach—the Ada Process Model. To capitalize on this new process, traditional software engineering activities have had to adapt. The system performance estimation activity was no exception. The performance model interacted with this new approach in a unique fashion. This uniqueness is not as a result of new modeling techniques, but because the Ada Process Model provides complementary information to the performance model as an indirect result of its approach. This gives the performance model something that prior projects have not had until after CDR, empirical software performance data.

## 2. THE Ada PROCESS MODEL

Figure 1 is an overview of a generic definition of an incremental development derived from experience on CCPDS-R [Royce 1990]. This figure shows how incremental development feeds the demonstration milestones and other project areas. Royce describes in his paper how code is developed early, through partial implementation of software functions. This provides early insight into the functionality of the design. The main objective of the Ada Process Model (APM) is early implementation of software. This concept is exemplified by the fact that of the five software builds on CCPDS-R, four are completed prior to System CDR.

## 3. CCPDS-R PERFORMANCE MODEL DEVELOPMENT

The CCPDS-R Performance Model (CPM) is exceptional in its use in concert with the APM and its interaction with the software testbed.

The purpose of the CPM is to: (1) investigate system design issues in advance of software implementation, (2) establish performance allocations to the CSCIs, (3) provide performance predictions for the operational system configuration as the design evolves, (4) provide a measure to evaluate the performance of capability demonstrations, and (5) provide insight for design tradeoffs and proposed system upgrades.

Figure 2 shows the relationship of the system performance model to a performance database. The performance database centrally houses performance information describing the system design. Contributions to this database include software engineering inputs, prototyping results, commercial or off-the-shelf (COTS) hardware and software performance data, operational performance data and testbed performance results.

### 3.1 Relationship of CPM to Program Testbeds

The CCPDS-R Performance Prediction/Measurement Process uses three performance sources: (1) the CPM simulating CCPDS-R hardware and software architecture, (2) a prototype system testbed (Host or target based), and (3) the operational system testbed, running on the target hardware environment.

The CPM is a simulation based upon a collection of transaction flow descriptions and a data base of performance characteristics of the processing performed by each transaction flow and the underlying hardware. (A transaction flow is the sequence of processing that occurs in response to some system stimulus such as an incoming sensor message or an operator input.)

The prototype testbed is a model of the processing that uses some stubbed versions of the CCPDS-R to top level tasks. These tasks perform the same message routing logic as their counterparts in the real system without necessarily performing the same computations. Instead, they may contain simple central processing unit (CPU) resource burning procedures that correspond to the current estimates of the resource requirements in the real system as defined by the software designers.

The operational system testbed contains the turned over software that is used to verify the functional and performance requirements at the Functional Qualification Test (FQT) milestone.

The CPM and the prototypes use a common data base of estimates of software resource requirements. They are updated with new design information as it becomes available. Consequently, both the fidelity and accuracy of these techniques increase throughout the system development process.

The estimation methods complement each other. The CPM provides early visibility into the performance of the design. The prototype testbed provides a target environment for executing prototyped software. This testbed can then be used to provide data for the CPM to refine the model's fidelity. The target environment also provides empirical system data to the model (e.g., Operating System Overhead Factors).

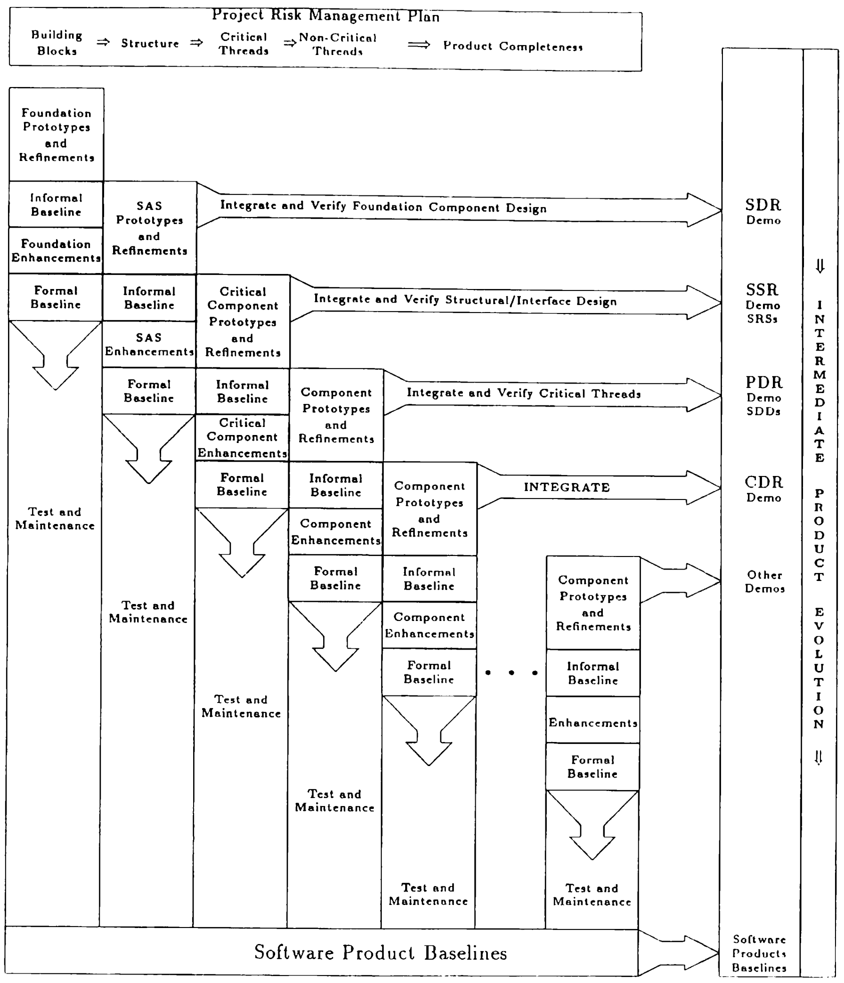
Project Risk Management Plan
Building Blocks ⇒ Structure ⇒ Critical Threads ⇒ Non-Critical Threads ⇒ Product Completeness
Foundation Prototypes and Refinements
Informal Baseline
Foundation Enhancements
SAS Prototypes and Refinements
Integrate and Verify Foundation Component Design
SDR Demo
Formal Baseline
Informal Baseline
SAS Enhancements
Critical Component Prototypes and Refinements
Integrate and Verify Structural/Interface Design
SSR Demo SRSs
Formal Baseline
Informal Baseline
Critical Component Enhancements
Component Prototypes and Refinements
Integrate and Verify Critical Threads
PDR Demo SDDs
Formal Baseline
Informal Baseline
Component Enhancements
Component Prototypes and Refinements
INTEGRATE
CDR Demo
Formal Baseline
Informal Baseline
Component Enhancements
Component Prototypes and Refinements
Other Demos
Formal Baseline
Informal Baseline
Enhancements
Formal Baseline
Test and Maintenance
Software Product Baselines
Software Products Baselines
INTERMEDIATE PRODUCT EVOLUTION

**Figure 1.** Incremental Development Under The Ada Process Model

The operational CCPDS-R (having built-in performance monitoring and report generation capabilities) supplements the two performance estimation methods. In the latter phases of the CCPDS-R development cycle, the operational testbed becomes the primary source of definition of CCPDS-R performance characteristics.

## 3.2 Model Results

The CPM produces results typical of most ADPE performance models. CPU utilization percentages, system responsiveness, queue statistics, message load statistics, and system specific measures are all provided as outputs of the CPM. The CCPDS-R program conducts monthly reviews for software and all systems engineering. The performance modeling group has an opportunity at each of these monthly reviews to discuss results of the CPM or issues concerning the design. This has proven to be an excellent method for communicating model results and raising issues. Formal reviews are also used to communicate model results to the customer. Each review milestone (SSR, IPDR, PDR, CDR) has required a presentation on the performance modeling analysis and results.

## 4. PROCESS MODEL IMPACTS ON PERFORMANCE PREDICTION

Most conventional development models predict requirement satisfaction as shown in Figure 3. Estimates of needs increase as the design matures. When the software is actually built and integrated at FQT, the software performance typically is significantly higher than the requirement. Use of the APM results in a different trend [Royce 1990]. Software development begins early in the project. If noncompliance with requirements is a significant risk it is indicated at a point in the project where analysis and optimization is more efficient [Boehm 1981 and Boehm 1985]. As shown in the figure, there are peaks and valleys in the graph. The peaks relate to demonstrations (e.g., PDR Demo). As a capability demonstration is integrated and prepared for review, performance optimization is a lower priority than working functionality. After the demonstrations the integration team then analyzes the code and resolves performance bottlenecks and inefficiencies. Subsequent adjustments cause the valleys. The use of capability demonstrations trades early functionality for performance immaturity. This is a desirable approach since performance enhancements are simpler and cheaper to make early in the design.

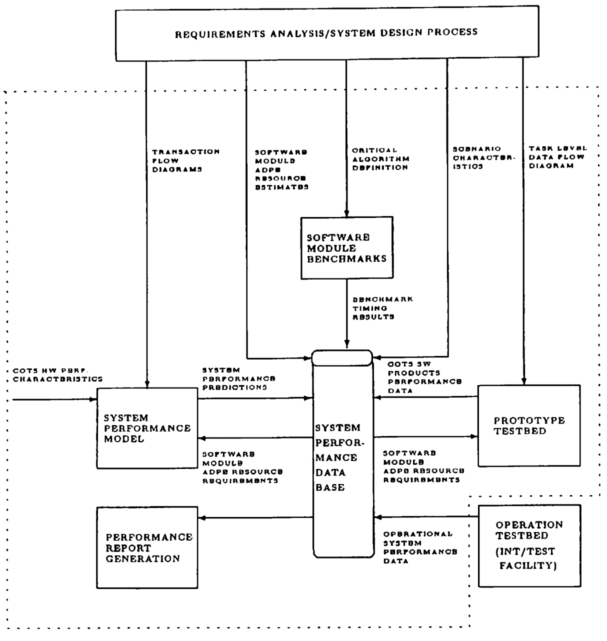
REQUIREMENTS ANALYSIS/SYSTEM DESIGN PROCESS
TRANSACTION FLOW DIAGRAMS
SOFTWARE MODULE ADPE RESOURCE ESTIMATES
CRITICAL ALGORITHM DEFINITION
SCENARIO CHARACTER-ISTICS
TASK LEVEL DATA FLOW DIAGRAM
SOFTWARE MODULE BENCHMARKS
BENCHMARK TIMING RESULTS
COTS HW PERF. CHARACTERISTICS
SYSTEM PERFORMANCE PREDICTIONS
COTS SW PRODUCTS PERFORMANCE DATA
SYSTEM PERFORMANCE MODEL
SYSTEM PERFOR-MANCE DATA BASE
PROTOTYPE TESTBED
SOFTWARE MODULE ADPE RESOURCE REQUIREMENTS
SOFTWARE MODULE ADPE RESOURCE REQUIREMENTS
PERFORMANCE REPORT GENERATION
OPERATIONAL SYSTEM PERFORMANCE DATA
OPERATION TESTBED (INT/TEST FACILITY)

**Figure 2.** CCPDS-R Performance Prediction/Measurement Process

The CPM helps this process by providing a benchmark for evaluation of the code. Without this benchmark, there would be little basis for where performance enhancement and analysis should begin and when performance optimization should stop.

The CPM also gives information on the performance of code that is missing from the demonstrations. The CPM provides the expectations against which the real software performance is judged.

## 5. REAL WORLD EXPERIENCE

The CPM has guided the CCPDS-R design in a number of areas. Resolution of identified issues are incorporated into the CPM and assessed for impact to the system performance. This relationship has been instrumental in early risk identification and mitigation.

1. Demonstration success criteria
2. Inter-Task communication performance
3. Display performance
4. Alarm processing performance
5. Design parameter determination

### 5.1 Functional and Performance Demonstrations

The CCPDS-R testbed is used to build demonstrations for government reviews. The CPM is used to establish pass/fail criteria on the performance of demonstrations. Typical criteria for these demonstrations include process CPU utilization percentages taken as output from the CPM. Failure to watch these percentages, within a tolerance, generates an action item to explain the discrepancy update the model and/or identify a design issue for resolution. At the CCPDS-R System PDR Capability Demonstration, the CPM identified fourteen pass/fail criteria. Of the fourteen criteria, twelve failed. Most of the failed criteria has resulted in modifications to the design. Some issues required updates to the CPM as well (e.g., display processing estimates and data recording estimates). A plot of the performance of the processor utilization during the PDR demo versus the predicted utilization from the CPM is shown in Figure 4. The CDR capability demonstration results are shown in Figure 5. The results reflect the performance of the processor without benefit of any analysis to improve the performance. There is a significant difference between the CPM and the demonstration. As with the PDR demonstration, action items have been generated to resolve discrepancies between the CPM and the demonstration results.

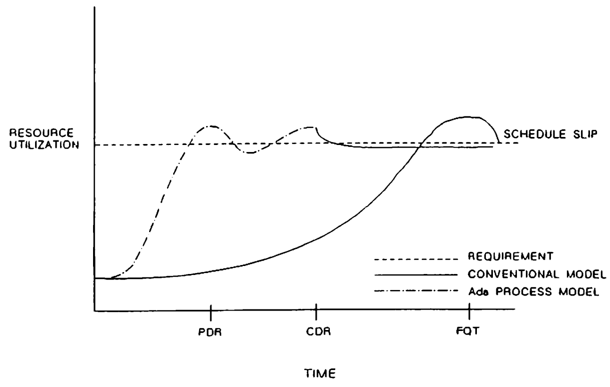

**Figure 3.** Software Development Progress

The role of the CPM in the process model approach is to reap the benefits of early capability demonstration results. The performance data from these demonstrations combined with designer's estimates provides a prediction tool that estimates performance trends prior to completion of all the software components.

### 5.2 ITC Performance

One of the major components of CCPDS-R is the Network Architecture Services (NAS) software. Contained in this software are the components that provide the Inter-Task Communication (ITC) functions. This software provides the features of Ada involved with tasking. Consistent with the Ada process model, this software was developed early in the program to minimize risk and to allow its use by the application software. Due to the use of this software by all other components in the system, ITC performance has been the focus of a great deal of analysis from the beginning of the program until present. There are three levels of ITC. Level 1 ITC involves communication between two tasks within one VAX/VMS process. Level 2 ITC interfaces two task in different processes. Level 3 ITC permits communication between tasks in different processors within the same subsystem (network). For each of the three ITC levels, the communication can be buffered or unbuffered. A more detailed description of the functionality of this software is well documented in [Royce 1989]. The significant factor in the process model approach for development of this software was that the functions provided by ITC were required as a basis for use by all other software development areas. Further, these areas were able to use this software in the development, and improve on the performance and functionality through System CDR. This area of software development best exemplifies the benefits of the Ada process model.

As the project progressed, increased functionality was required of the ITC software. The performance impacts of these changes were captured in standalone throughput testing. This testing provided a measure of the number of messages/second achieved by each level of ITC. This information is reflected in Table 1. Each sample reflects a measure-

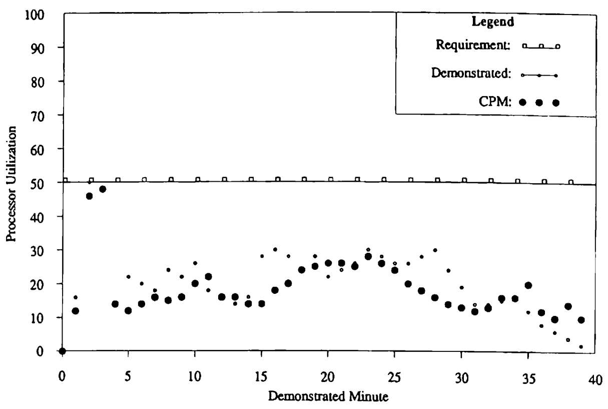

**Figure 4.** PDR Demonstration Processor Utilization versus the CPM Prediction

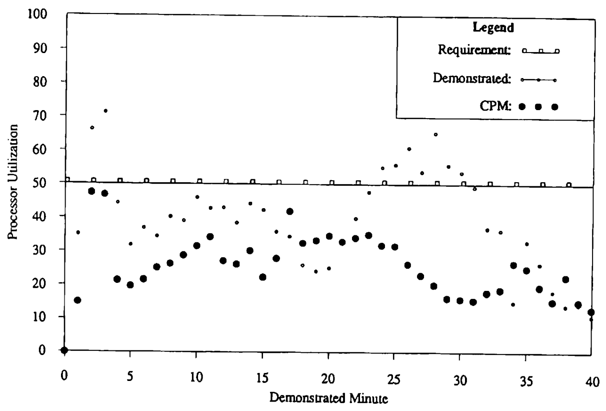

**Figure 5.** CDR Demonstration Processor Utilization versus the CPM Prediction

ment of the performance of the software. The CPM incorporated this information and provided feedback on the impact to the system for the given changes/improvements. The results of the CPM would then provide a measure to indicate if performance optimizations were adequate or if further improvements were required to meets the system performance requirements. The results of this process were reflected in allocations to each ITC level in terms of number of messages/second required by the software.

Table 1 shows throughput performance of ITC as design changes and new functionality were added to the software. The measurement in January, 1989 reflects a major change to the software. The next measurement in May, 1989 reflects performance enhancements made to overcome the performance degradation of the prior enhancements. The last column in Table 1 reflects the throughput requirements derived for each ITC level. These are the requirements that the developers used to measure the performance enhancements against. The values were derived from assessments of system performance impacts of the measured software on the system performance requirements using the CPM. The CPM provided the feedback necessary to indicate that further performance enhancements were not required.

**Table 1.** ITC Performance

| ITC Level | UnBuffered/ Buffered | Messages/Second | | | | |
|---|---|---|---|---|---|---|
| | | 9/88 | 12/88 | 1/89 | 5/89 | Allocation |
| 1 | UnBuffered | 2,078 | 2,080 | 1,702 | 1,855 | 1,800 |
| | Buffered | 8,861 | 8,000 | 6,172 | 8,251 | 8,000 |
| 2 | UnBuffered | 495 | 459 | 485 | 409 | 350 |
| | Buffered | 4,342 | 3,784 | 3,223 | 4,260 | 3,500 |
| 3 | UnBuffered | 250 | 267 | 282 | 376 | 300 |
| | Buffered | 3,556 | 3,429 | 2,920 | 3,520 | 3,000 |

## 5.3 Display Performance

The display performance requirements for CCPDS-R are very stringent. The system must produce a complex, graphic display within one second of an operator request. In the early phases of the project estimates to produce displays were made and incorporated into the CPM. As the prototyping effort began to produce the display software, display generation responsiveness was poor. All effort was then started to determine the source of this discrepancy. Table 2 lists the display generation responsiveness for a representative set of twelve displays that were tracked from SSR to CDR. Each sample relates to a point in time when measurements of display performance were taken. The first sample (Sample 0) relates to the SSR milestone and the last sample is CDR. A graph of the worst case display and the average display is shown in Figure 6. An improvement of more than 90% was made for the average display since SSR. The resultant times are within 10% of the predicted times from the CPM.

**Table 2.** Display Generation Responsiveness Over Time

| Preformatted Display Generation (average seconds) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISPLAY NAME | SAMPLE | | | | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Display 1 | 19.4 | 12.1 | 10.0 | 6.5 | 2.9 | 1.7 | 1.6 | 1.4 | 1.2 | 0.9 | 0.7 | 0.7 |
| Display 2 | 8.9 | 6.0 | 5.1 | 2.8 | 1.4 | 1.2 | 1.7 | 1.1 | 1.1 | 0.6 | 0.4 | 0.4 |
| Display 3 | 2.9 | 1.5 | 1.3 | 0.9 | 0.9 | 0.8 | 1.5 | 1.0 | 1.0 | 0.8 | 0.6 | 0.6 |
| Display 4 | 3.3 | 1.6 | 1.3 | 1.0 | 0.8 | 0.7 | 1.7 | 1.2 | 1.2 | 0.7 | 0.7 | 0.7 |
| Display 5 | 5.8 | 4.0 | 2.4 | 1.3 | 1.0 | 1.0 | 1.9 | 1.5 | 0.9 | 0.8 | 0.4 | 0.3 |
| Display 6 | 10.1 | 5.7 | 4.8 | 2.9 | 1.8 | 1.6 | 2.5 | 1.8 | 1.4 | 1.2 | 1.1 | 1.0 |
| Display 7 | 8.3 | 6.0 | 5.0 | 3.9 | 3.1 | 1.5 | 2.1 | 1.8 | 1.4 | 1.2 | 0.7 | 0.6 |
| Display 8 | 16.0 | 10.9 | 8.9 | 4.9 | 3.2 | 1.6 | 2.9 | 2.2 | 1.6 | 1.1 | 1.2 | 1.6 |
| Display 9 | 15.7 | 10.5 | 8.6 | 4.9 | 2.9 | 1.5 | 2.7 | 2.1 | 1.7 | 1.4 | 1.7 | 2.0 |
| Display 10 | 14.9 | 7.0 | 6.2 | 3.9 | 1.9 | 1.5 | 2.7 | 2.2 | 1.7 | 1.1 | 1.6 | 2.2 |
| Display 11 | 9.0 | 6.4 | 4.9 | 2.9 | 1.7 | 1.5 | 2.5 | 1.8 | 1.5 | 0.7 | 0.7 | 0.7 |
| Display 12 | 9.0 | 6.4 | 4.9 | 2.9 | 1.7 | 1.4 | 2.9 | 2.7 | 1.5 | 0.6 | 0.5 | 0.5 |
| Average | 9.6 | 6.5 | 5.3 | 3.2 | 1.9 | 1.3 | 2.2 | 1.7 | 1.4 | 0.8 | 0.9 | 0.9 |

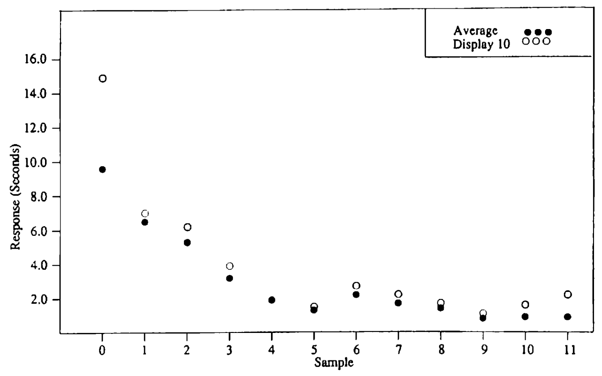

**Figure 6.** Display Generation Responsiveness Over Time

### 5.4 Alarm Processing

The CPM has also influenced the design in the area of the alarm processing function. When this function was modelled, an alarm overload became apparent. When first simulated, the alarm processing and acknowledgment functions saturated one of the processors. The assumption was that the CPM was in error. After closer examination, it was deemed that the CPM was correct and the processing was accurately modeled. it was apparent the operator could not possibly acknowledge all the alarms individually nor did they provide useful information after the first few were displayed. Since then, the CPM is being used to quantify the impact of different alarm filtering techniques on the operator as well as the ADPE. This effort also resulted in a redesign incorporating a window acknowledgement scheme. The redesign was implemented prior to CDR and evaluated by the customer.

### 5.5 Design Parameters

One of the advantages of an ADPE performance simulation is the early quantification of assumptions and identification of design weaknesses. During the CPM development, many assumptions were made to provide a complete model of the CCPDS-R because the performance model requires some form of data for all components modeled. The simulation will run without this data; however, lack of data provides easy identification of holes in the design. Initially, some of these were:

- Operator Interactions - How the operator interacted and reacted to the system.
- Subsystem to Subsystem Interface Loading - Definition of how message exchange would be accomplished to exchange the databases.
- Display Scenario Definitions Keyed to the Processing Load Scenario - Which displays are to be displayed as a function of the scenario used.
- Display Update Rates - Rates at which the displays would be updated.
- Task to Task Buffered Communication Parameters - Values to be used to meet the performance requirements of the system.
- Approaches for Handling Off-Nominal Conditions - Paths and functionality for handling conditions that occur during non-critical periods of the scenario.

Due to the process model and the early identification of these issues, the incorporation of the resolutions into the design could be made at a less costly point in the development.

## 6. CUSTOMER USE OF THE CPM

As part of the performance modelling activities, the CPM is given to the customer at regular intervals in the development. The CPM provides detailed flows of information through the system in a graphical form. These flows also contain processing delay information for given functions listed on the transaction flow. Customer use of the CPM has provided a mechanism for understanding the detailed system design early in the design process.

By manipulating the inputs to the CPM, the customer can then play "what if" games using the latest design information. This gives the customer the basis of the design needed to gather more specific information from the developers.

## 7. SUMMARY AND CONCLUSIONS

The CCPDS-R program has maximized the use of an ADPE performance model during all phases of a systems design. It also provided an approach for its use in concert with the Ada Process Model. With the effective use of the CPM and the Ada Process Model you can maximize the efficiency of both efforts. The CPM is a complementary tool in the Ada Process Model to build a computer system which meets its requirements.

## REFERENCES

Boehm, B.W. (1981), *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ.

Boehm, B.W. (1985), "The Spiral Model of Software Development and Enhancement," *Proceedings of the International Workshop on the Software Process and Software Environments*, Coto de Caza, CA.

Royce, W.E. (1989), "Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-Task Networks: Network Architecture Services (NAS)," *TRI-Ada Proceedings*, Pittsburgh, PA.

Royce, W.E. (1990), "TRW's Ada Process Model for Incremental Development of Large Software Systems." Submitted to *12th International Conference on Engineering*, Nice, France.