EXHIBIT II SACCHARIDE DISTRIBUTION VS. TIME

EXHIBIT III RAIL CAR LOADINGS

LOADINGS FOR DAY 4

CAR NO	RAW TIME	TIME FNSHD	DS LBS	% DS	% DEX	% MALT	% MALTRI	% UPS	% DE	PRODUCT	BE'(100F)
1	73.75	1.75	161941.7	79.8	8.1	30.9	24.4	36.5	43.0	OS	42.6
2	74.79	2.79	80539.8	79.3	9.5	38.7	22.0	29.8	46.6	OS	42.2
3	77.05	5.05	161766.1	79.3	10.1	41.8	21.1	27.1	48.0	3	42.2
4	79.09	7.09	80788.5	80.1	9.7	42.8	21.7	25.8	48.3	3	42.6
5	80.80	8.80	81029.0	80.0	9.4	43.3	22.2	25.1	48.3	3	42.6
6	83.65	11.65	160301.5	80.0	9.3	43.3	22.4	25.0	48.3	3	42.5
7	84.69	12.69	80543.4	79.8	9.4	42.7	22.4	25.6	48.1	3	42.4
8	86.29	14.29	81148.1	79.9	9.3	42.5	22.4	25.8	48.0	3	42.5
9	88.70	16.70	162201.7	79.8	9.3	42.6	22.5	25.5	48.0	3	42.5
10	90.00	18.00	80596.1	79.8	9.4	43.0	22.8	24.8	48.3	3	42.4
11	93.25	21.25	161248.9	80.0	9.4	42.9	22.7	25.0	48.3	3	42.5
12	94.96	22.96	80201.9	80.0	9.1	42.6	22.2	26.1	47.8	3	42.6

LOADINGS FOR DAY 5

CAR NO	RAW TIME	TIME FNSHD	DS LBS	% DS	% DEX	% MALT	% MALTRI	% UPS	% DE	PRODUCT	BE'(100F)
1	96.10	0.10	80393.0	80.0	9.1	42.5	22.0	26.4	47.7	3	42.6
2	98.75	2.75	160594.0	79.9	9.4	42.8	21.9	26.0	48.0	3	42.5
3	101.40	5.40	160348.2	79.9	10.0	43.3	21.7	25.0	48.7	3	42.4
4	105.47	9.47	160544.2	79.9	23.7	40.5	17.5	18.2	57.8	OS	42.1
5	108.09	12.09	161900.3	79.8	38.2	37.5	13.0	11.3	67.3	4	41.7
6	110.70	14.70	161478.7	79.6	39.2	35.9	12.5	12.5	67.4	4	41.6
7	113.65	17.65	161649.6	80.0	39.4	35.4	12.3	12.9	67.4	4	41.8
8	115.25	19.25	81027.6	80.0	39.2	35.3	12.1	13.3	67.2	4	41.8
9	116.85	20.85	80996.9	80.0	39.2	35.3	12.1	13.4	67.1	4	41.8
10	118.45	22.45	80959.3	80.0	39.3	35.4	12.4	12.9	67.3	4	41.8
11	120.00	24.00	80043.9	80.0	39.5	35.8	13.4	11.2	67.9	4	41.8

LOADINGS FOR DAY 6

CAR NO	RAW TIME	TIME FNSHD	DS LBS	% DS	% DEX	% MALT	% MALTRI	% UPS	% DE	PRODUCT	BE'(100F)
1	121.07	1.07	80370.8	80.0	39.6	36.0	14.0	10.5	68.1	4	41.8
2	123.65	3.65	161196.2	79.8	39.4	35.9	13.8	10.9	67.9	4	41.7
3	125.71	5.71	160120.1	79.3	39.0	35.7	12.7	12.6	67.2	4	41.4
4	126.75	6.75	80604.3	79.6	39.0	35.6	12.4	13.0	67.1	OS	41.6
5	130.42	10.42	160698.1	79.6	39.3	30.1	7.6	23.0	64.4	1	41.7
6	132.70	12.70	161187.6	79.7	39.6	28.8	6.3	25.3	63.9	1	41.8
7	135.05	15.05	160212.2	80.0	40.6	29.7	6.5	23.2	65.1	1	41.9
8	137.15	17.15	161117.0	79.8	40.4	29.5	5.8	24.3	64.7	1	41.8
9	139.80	19.80	162236.8	79.9	39.9	29.0	4.5	26.6	63.8	1	41.9
10	140.92	20.92	80962.2	80.0	39.8	29.1	5.3	25.8	63.9	1	41.9
11	143.45	23.45	160288.1	79.9	39.6	29.1	6.8	24.5	64.1	1	41.9

LOADINGS FOR DAY 7

CAR NO	RAW TIME	TIME FNSHD	DS LBS	% DS	% DEX	% MALT	% MALTRI	% UPS	% DE	PRODUCT	BE'(100F)
1	146.02	2.02	161320.8	79.5	39.3	28.9	6.5	25.2	63.7	1	41.7
2	148.60	4.60	161216.5	79.8	39.1	28.6	6.0	26.3	63.3	1	41.9
3	149.64	5.64	80385.5	79.8	39.7	29.1	5.6	25.6	63.9	1	41.8

COMPUTER SYSTEM SIMULATION WITH GASP IV

Gain Wong
General Electric Company, Sunnyvale, California

SIMTRAN is a process oriented computer system simulator. SIMTRAN employs a preprocessor to allow the analyst to take a process oriented view of the system he is attempting to model. A process oriented view is superior to the usual event oriented view, when modeling complex computer systems, where multi-level modeling and easy modification of the model are desired.

SIMTRAN was developed from GASP IV, a general purpose FORTRAN-based simulation language. All of the simulation modeling and reporting capabilities available in GASP IV are available in SIMTRAN. In addition, SIMTRAN provides facilities for modeling process and resource management. These facilities are very helpful in constructing simulation models of computer systems because a large part of the behavior of computer systems depends on process and resource management strategies.

MODELING CAPABILITIES

Computer simulation is a modeling technique where a computer program is written whose behavior resembles that of the system which is being modeled. The more closely the behavior of the simulation program resembles the behavior of the real system, the better is the simulation model. A simulation language is a tool which facilitates the development of simulation programs, and aids in exercising them.

GASP IV is a general purpose FORTRAN-based simulation language. GASP IV provides the user with an event oriented view of his system, facilities for collecting statistics about the simulation model and generating reports, and the power and flexibility of the procedure oriented programming language FORTRAN IV.

GASP IV may be successfully utilized to perform computer system simulation. The modeling of computer systems would be significantly aided however by 1) allowing the user to take a process oriented view, and 2) providing facilities for modeling process and resource management.

GASP IV imposes an event oriented view of the system being modeled. This view tends to be single level since all state changes of the system, occurring in various levels of detail, are represented by events at one level. A process oriented view lends itself to multi-level modeling since process descriptions allow the analyst to describe a process in terms of its activities and the conditions under which the process proceeds from one activity to another, without having to explicitly sequence events. An activity of a process can easily be a process with its own activities.

With a process oriented view, the system is viewed dynamically as a collection of interacting processes, with the interactions controlled and coordinated by the occurrence of events. The advantages of this view are many. One advantage is that a process oriented model is many times a more natural way to express the structure of a system. Another advantage is that the user does not have to define and keep track of the events which signal state changes in the system. Instead, event definition and sequencing is done by the simulator, and is invisible to the user. A further advantage is that process orientation automatically provides for process structuring. A process, with all of the event definition and sequencing implied by it, can easily be a subprocess of another process. Process structuring is very difficult to obtain with strictly an event oriented view.

GASP IV does not provide explicit facilities for modeling process and resource management, because it is a general purpose simulation language. Such facilities, however, would be very valuable for the user who wishes to model computer systems. SIMTRAN provides both process and resource management facilities. The process management facilities are useful because of the process oriented view allowed by SIMTRAN. The resource management facilities are important because of their heavy use in the modeling of computer systems.

Five process management facilities are available in SIMTRAN. These are 1) INIT - to initiate a process, 2) ACTIV - to activate a process which has previously been initiated, 3) DELAY - to cause the current process to be suspended and re-activated after a specified time interval, 4) SUSPEND - to cause the current process to be suspended, and 5) TERM - to cause the current process to be terminated.

A SIMTRAN process is an incarnation of a SIMTRAN process description. Zero, one or more processes may exist for a given process description. Each incarnation of a process description is an independent entity; process-local memory is maintained by the SIMTRAN system. Processes are initiated by another process executing an INIT, and they terminate when they execute a TERM. Processes may execute ACTIVs to coordinate the execution of other processes, and they may execute DELAYs and SUSPENDs to coordinate their own execution.

Five resource management facilities are available in SIMTRAN. These are 1) CREATE - to create a descriptor of a resource of a specified type and with a specified initial allocation, 2) RQST - to request a unit of a previously created resource, 4) STATUS - to obtain status information about a previously created resource, and 5) DESTR - to destroy the descriptor of a previously created resource; all references to that resource will no longer be valid.

A SIMTRAN resource descriptor represents a real resource. It may be dynamically created and destroyed. Processes may invoke RQST, RLSE and STATUS calls on previously created resources. The SIMTRAN system will allocate and de-allocate resource units according to the type of the resource. Further, the SIMTRAN system will keep statistics on the utilization of all resources.

SIMTRAN is capable of modeling three types of resources. Message (M) type resources are consumable resources, in that they are produced by one process and consumed by another. Processes produce message units by RLSEing them, and they consume message units by RQSTing them. A process that RQSTs a unit of a message resource for which no resource units currently exist is SUSPENDed, put on a priority queue, and ACTIVated when a message unit becomes available.

Facility (F) type resources are serially reusable resources. This means that the resource has a finite, fixed number of units, each of which may be free (unallocated) or allocated. Units of facility type resources are allocated to a process when the process executes a RQST, and a resource unit is available for allocation. Facility type resource units are deallocated when a process which was previously allocated a resource unit executes a RLSE. A process that RQSTs a unit of a facility resource for which no resource units are currently free is SUSPENDed, put on a priority queue, and ACTIVated when a resource unit becomes free.

Storage (S) type resources are also serially reusable resources. They are different from facility type resources in the way in which resource units are allocated. Facility type resources are allocated a unit at a time, where each unit is equivalent. Storage type resource units are numbered; they are allocated a chunk at a time, where each chunk consists of a specified number of consecutively numbered storage resource units. If a process RQSTs a chunk of a storage resource and no sufficiently large chunk of the storage resource is free, then the process is SUSPENDed, put on a priority queue, and ACTIVated when a large enough chunk becomes free.

SIMTRAN DESIGN AND IMPLEMENTATION

SIMTRAN allows the analyst to develop a process oriented computer simulation program within the FORTRAN environment. In order to allow process oriented modeling, SIMTRAN must provide 1) routines in the run time executive to handle process initiation, synchronization, sequencing and termination, 2) a mechanism which allows a process which is executing within a FORTRAN subroutine to be suspended and later reactivated where it left off, and 3) a mechanism which allows different processes, executing within the same FORTRAN subroutine, to access variables local to the process.

SIMTRAN processes are represented by a process description and a transaction number. A process description is simply a FORTRAN subroutine. A transaction number is a pointer to a set of data representing the values of the process-local attributes. The process management routines provided by SIMTRAN will now be discussed in more detail.

1. CALL INIT (T, process #, activity #, priority, transaction #)

   This routine performs process initiation; storage is allocated for the attributes of the new process and a transaction number is returned pointing to the location of these attributes. This process is then scheduled to start at the current simulated time plus T.

   The process # identifies the process description for the process to be initiated. Processes are divided into activities; SIMTRAN provides facilities for a process to control its own progression from one activity to the next (DELAY and SUSPEND) and to control the progression of other processes from one activity to the next (ACTIV). The priority of a process is used for scheduling its execution.

2. CALL ACTIV (T, process #, activity #, priority, transaction #)

   This routine performs process synchronization; the process specified by the process # and transaction # is scheduled to occur at the current simulated time plus T. The activated process is restarted at the specified activity and with the specified priority.

3. CALL DELAY (T)

   This routine performs process sequencing; the current process becomes inactive and is scheduled to be reactivated at the current simulation time plus T.

4. CALL SUSPEND

   This routine also performs process sequencing; the current process becomes inactive. It will remain inactive until activated by another process.

5. CALL TERM

   This routine performs process termination; storage is de-allocated for the attributes of the current process.

SIMTRAN processes consist of a number of activities. An activity is the unit by which the SIMTRAN process management routines schedule processes. An activity number can be associated with each activity of a process. If a computed go to statement, indexed by activity number, is placed at the beginning of the FORTRAN subroutine representing the process, then we have a means of allowing a process to become inactive, and then be reactivated where it left off. Process n executing in activity m can schedule itself to be reactivated at a later time by making an entry for process n, activity m+1 in the event list, and then returning control to the run time executive. When the process is later reactivated, the computed go to statement will cause the process to resume with the proper activity.

It is clear that a new activity of a process is defined at each point in the process where the process could become inactive. The determination of the activities of a process and the assignment of activity numbers may be done explicitly by the user, or they may be performed by a preprocessor. The latter alternative has been taken in order to foster structured programming practices.

The simulation language constructs provided by SIMTRAN are as follows:

1. PROCESS name

   This construct identifies the beginning of a process description. The preprocessor generates a FORTRAN subroutine statement for a subroutine with the specified name. Further, the preprocessor generates a labeled common declaration for the SIMTRAN internal variables and process local variables. The SIMTRAN internal variables include the current simulation time, the current process number, the current activity number, the current priority, and the current transaction number. The process local variables are for the process local attributes.

2. BEGIN

   This construct identifies the first activity of the process. An unconditional go to statement is generated. The destination of this go to statement will be a computed go to statement, indexed by activity number, at the end of the subroutine. The reason the computed go to statement is placed at the end of the subroutine is the constituent activities of the process are not known until then. A statement label is then generated by the preprocessor. This statement label will be associated with activity number 1.

3. CALL DELAY (T)

   This construct causes code to be generated so that (a) the current activity number is updated by incrementing it by one, (b) the process management routine to reactivate the current process at the current simulation time plus T is called, and (c) the process then returns to the run time executive. Further, a new statement label is generated by the preprocessor. This statement label will be associated with the updated activity number.

4. CALL SUSPEND

   This construct causes code to be generated so that the current activity number is updated by incrementing it by one. Further, a new statement label is generated by the preprocessor. This statement label will be associated with the updated activity number.

5. CALL RQST (resource #, parameter)

   This construct causes code to be generated so that the current activity number is updated by incrementing it by one. A new statement label is then generated by the preprocessor. This statement label will be associated with the updated activity number. Next, code is generated so that (a) the resource management routine to request a unit of the specified resource is called, and (b) the value returned by the resource management routine is tested: if the value returned indicates a request failure, then the process returns to the run time executive. In this case, the current process has

been entered on the waiting list associated with the requested resource. When the requested resource becomes available, the requesting process will be reactivated. If the value returned indicates a request success, then the process continues with the next activity.

6. END

This construct identifies the end of a process description. A subroutine call to the process management routine for terminating processes is generated. A statement label whose statement number matches the destination of the unconditional go to statement generated for the BEGIN construct is now generated. Finally, the computed go to statement, indexed by activity number, with the statement numbers associated with each of the activities of the process is generated, followed by the FORTRAN END statement.

SIMTRAN provides the means for processes to maintain process local variables. This is accomplished by dynamically allocating storage for these variables when a process is initiated. When a process is activated by the run time executive, the variables local to the process are copied from the dynamically allocated storage area into a common block labeled LOCAL. This facilitates access to these local variables. When a process returns control to the run time executive, the current contents of the common block labeled LOCAL are copied back into the proper place in the dynamically allocated storage area.

The resource management routines provided by SIMTRAN enable the user to easily model the scheduling and allocation of resources in a computer system. The resource management routines provided by SIMTRAN will now be discussed in more detail.

1. CALL CREATE (resource type, resource name, resource #, parameter)

This routine performs resource creation. A descriptor is allocated for the resource being created and a resource number is returned pointing to this descriptor. The resource type may be message type, facility type, or storage type. The resource name is a name which is to be associated with this resource. The parameter specifies the number of resource units initially available.

2. Value = RQST (resource #, parameter)

This routine will allocate a unit of the specified resource to the requesting process if possible. If not, the requesting process will be placed on the waiting list associated with the requested resource. The resource type of the requested resource determines the allocation strategy that is taken. The parameter normally specifies the number of resource units requested.

The user writing his SIMTRAN program normally writes CALL RQST (x,y). The preprocessor translates this to ANS = RQST (x,y) so that code may easily be generated to test the value returned from RQST. The value returned is 1 for request success, and Ø for request failure.

3. CALL RLSE (resource #, parameter)

This routine will de-allocate a unit of the specified resource for facility and storage type resources, and produce a unit of the specified resource for message type resources. If any processes have been entered on the waiting list associated with the released resource, then the highest priority process is reactivated. The parameter normally specifies the number of resource units released.

4. CALL STATUS (resource #, parameter)

This routine returns status information concerning the specified resource. The number of free resource units of the specified resource is normally returned as the parameter.

5. CALL DESTR (resource #)

This routine performs resource destruction. The descriptor for the specified resource is de-allocated.

MODELING WITH SIMTRAN

SIMTRAN is used by the analyst to build a computer simulation model of the computer system that the analyst is studying. The analyst must understand both the static structure, and the dynamic behavior of the system he is attempting to model. Further, he must be able to visualize his system in terms of processes cooperating with each other to process data and processes competing with each other for resources.

A brief description of a SIMTRAN model of a distributed transaction processing system will now be presented. A transaction processing system is a computer system where a number of operators sit at terminals and every so often enter commands. These commands cause application tasks to begin execution. A distributed transaction processing system consists of a network of transaction processing systems. An operator sitting at a terminal connected to one computer system may enter commands which cause application tasks to begin execution at another computer system.

The processes in this system are (1) the operators who enter commands, and (2) the application tasks which execute commands. We may write a general process description for an operator, parameterized by two things - the computer system the operator is connected to, and the mix of commands that the operator is to enter. This process description will model the behavior of the operator by initiating various application tasks every so often. We may write a different process description for each application task. These process descriptions will model the behavior of the application tasks as they request computer resources to accomplish their processing, and later release them.

The resources in this system are (1) the central processor in each computer system, (2) the central memory in each computer system, and (3) the input/output channel in each computer system. The application tasks initiated by the operator processes will compete for these resources. SIMTRAN will gather statistics about the throughput of application tasks for a given amount of resources, and about the utilization of resources for a given mix of application tasks.

The model of a distributed transaction processing system presented above is a simple one, but it shows how the analyst would go about visualizing his system in terms of processes and resources. Further, the simple model is capable of providing useful insights about the gross behavior of such a system. A more detailed model might include such things as the communication system over which operator commands are transmitted, an application task manager which schedules application tasks on a priority basis, and a data base manager which controls access to the data base by providing concurrency checking. The model that the analyst will finally build depends on what aspects of the system he is interested in studying.

The SIMTRAN user is not limited to a process oriented view; the next event oriented view of GASP IV is still available to him. As long as the analyst is doing discrete simulation, however, the process oriented view of SIMTRAN will allow the analyst to use all of the modeling and report generation capabilities of GASP IV. It is only when the analyst wishes to do continuous simulation that the analyst must leave the process oriented view.

The analyst who wishes to perform combined discrete/ continuous simulation may easily interface the state event modeling of GASP IV with the process oriented view of SIMTRAN. When a GASP IV state event occurs, a SIMTRAN process may be initiated to perform some processing, or a message may be sent to an already existing SIMTRAN process.

Table 1 SIMTRAN Simulation Constructs

```
PROCESS name
BEGIN
CALL DELAY (T)
CALL SUSPEND
CALL RQST (resource #, parameter)
END
```

Table 2 SIMTRAN Process and Resource Management Facilities

process management

- INIT (T, process #, activity #, priority, transaction #)
- ACTIV (T, process #, activity #, priority, transaction #)
- DELAY (T)
- SUSPEND
- TERM

resource management

- CREATE (resource type, resource name, resource #, parameter)
- RQST (resource #, parameter)
- RLSE (resource #, parameter)
- STATUS (resource #, parameter)
- DESTR (resource #)

```
PROCESS name

     (declarations)

BEGIN

     (SIMTRAN, GASP IV and FORTRAN statements)

END
```

Figure 1 Structure of a SIMTRAN Process Description

SIMTRAN CONSTRUCT		FORTRAN STATEMENTS
PROCESS name		SUBROUTINE name
		COMMON/LOCAL/LOCAL (25)
		INTEGER ANS, ACTVN
		EQUIVALENCE (ACTVN, LOCAL (3))
BEGIN		GO TO N
	M	CONTINUE
CALL DELAY (T)		ACTVN = next activity #
		CALL DELAY (T)
		RETURN
	M	CONTINUE
CALL SUSPEND		ACTVN = next activity #
		RETURN
	M	CONTINUE
CALL RQST (X,Y)		ACTVN = next activity #
	M	CONTINUE
		ANS = RQST (X,Y)
		IF (ANS.EQ.Ø) RETURN
END		CALL TERM
	N	CONTINUE
		GO TO (M1, ..., Mn), ACTVN
		END

Figure 2 Implementation of SIMTRAN Constructs

REFERENCES

1. Blunden, G. P. and H. S. Krasnow, "The Process Concept as a Basis for Simulation Modeling", Simulation, Vol 9, No. 2 (Aug 1967).

2. Boyse, J. W. and David R. Warn, "A Straightforward Model for Computer Performance Prediction", Computing Surveys, Vol 7, No. 2 (June 1975).

3. Dahl, O. J. and K. Nygaard, "SIMULA - An ALGOL-based Simulation Language", CACM, Vol 9, No. 1 (Sep 1966).

4. Dewan, P. B., Donaghey, C. F. and J. B. Wyatt, "OSSL - A Specialized Language for Simulating Computer Systems", Proc AFIPS SJCC, Vol 40, 1972.

5. Gordon, Geoffrey, System Simulation. Englewood Cliffs, N. J.: Prentice Hall, Inc., 1969.

6. Howard, V. J., "A Model to Investigate Interprocess Communication in a Distributed Computer System", IFAC-IFIP Workshop on Real Time Programming, Mar 1974.

7. Klernrock, Leonard, "Resource Allocation in Computer Systems and Computer-Communication Networks", IFIP Conference on Information Processing, Stockholm, Aug 1974.

8. MacDougall, M. H., "Computer System Simulation: An Introduction", Computing Surveys, Vol 2, No. 3 (Sep 1970).

9. MacDougall, M. H. and J. Stuart McAlpine, "Computer System Simulation with ASPOL", Symposium on The Simulation of Computer Systems, Gaithersburg, Md., Jun 1973.

10. Nielsen, Norman R., "ECCS: An Extendable Computer System Simulator", Third Conference on Applications of Simulation, New York, Dec 1969.

11. Pritsker, A. Alan B., The GASP IV Simulation Language. New York: John Wiley & Sons, 1974.

12. Saleeb, Shafeek J. and Mokhtar B. Riad, "Use of Simulation for Operating System Optimization", IFIP Conference on Information Processing, Stockholm, Aug 1974.

13. Shaw, Alan, Weiderman, Nelson, Andrews, Gregory, Felayn, Mary-Beth, Rieber, John, and Gain Wong, "A Multiprogramming Nucleus with Dynamic Resource Facilities", Software - Practice and Experience, Vol 5, No. 3, 1975.

14. Walden, David C., "A System for Interprocess Communication in a Resource Sharing Computer Network", CACM, Vol 15, No. 4, Apr 1972.

15. Weiderman, N. H., Synchronization and Simulation in Operating System Construction, PhD Thesis, Computer Science, Cornell University, Technical Report 71-102, 1971.

16. Zurcher, F. W. and B. Randell, "Iteration Multilevel Modeling - A Methodology for Computer System Design", Proceedings of the IFIP Congress, Edinburgh, Scotland, Aug 1968.