Mobile Communications System Simulator Development using Structured Analysis

Jarno M. A. Tanskanen[1] and Matti J. Rintamäki[2]
Signal Processing Laboratory
Helsinki University of Technology
P.O.Box 3000, FIN-02015 HUT, FINLAND
E-mail: [1]jarno.tanskanen@hut.fi, [2]matti.rintamaki@hut.fi

Abstract-**In this paper, Structured Analysis (SA) is employed in mobile communications system simulation software development. Programming via SA provides a well-defined and easily maintainable structure for the software in a graphical form, also revealing any hidden dependencies in the software. Thus, the resulting simulator is more reliable and maintainable over its whole lifecycle. With modern SA tools, documentation can be automatically updated. Also, a mobile communications network simulator "Netsim", developed at Helsinki University of Technology (HUT) since 1994, is described in the paper.**

I. Introduction

In this paper, two main points are considered; firstly, structured analysis (SA) method [1][2][3] is overviewed in the context of software development, and secondly, an efficient mobile communications network simulator structure is presented using SA. The benefits and drawbacks of SA are reviewed in this context, and the method is employed in modeling a mobile communications system simulation software called "Netsim", which has been used for simulations presented in [4][5][6][7]. Netsim is being developed in Communications Laboratory of HUT, part of the updating work also being done within Signal Processing Laboratory, HUT.

Development management is of great importance in any software project. This is especially true in an inhomogeneous research environment where several individual researchers and groups are working on, and with, the same simulation software package over several years. Thus, increasing amount of effort has to be put into verification and documentation of the software. The motivation of this work lies in easing the further development and documentation of Netsim, and in presenting the structure of Netsim itself. This paper is especially intended to promote *creating and maintaining scientifically trustworthy simulation programs*, but may also be used as a basis for structural analysis of most any project or organization structure.

Netsim communications system simulator development started in 1994 with a Matlab based communications system simulator [4]. At that time, the basic structure of the simulator was laid down. Originally, Netsim was based on the CODIT test bed [4], and it is currently being updated to correspond to a 3rd generation WCDMA standard. This update involves modifications to most of the simulator functions, and is a driving force for the SA presented in this paper. Meanwhile, individual research groups have been further developing Netsim by adding functions, such as adaptive antennas, and WCDMA power control functions, in order to utilize Netsim in their own fairly specific research fields.

In this paper, clarity of the presentation of Netsim has been stressed; as a result, this presentation could not contain all the SA elements and strictly formatted definitions necessary for automatic program code generation directly from the SA model. Also, it is impossible to describe a network simulator in full detail within this paper, unfortunately. Instead of strict C language, pseudo code has been employed. Still, the treatment given in this paper can serve as a starting point for creating an SA model of a communications system simulator even for automatic code generation.

This paper is organized as follows; in Chapter II, a short introduction to structured analysis is given, SA model of Netsim is described in Chapter III, and conclusions are stated in Chapter IV.

II. Introduction to Structured Analysis (SA)

A. Benefits and Drawbacks of SA

Graphical interface of a modern SA tool offers programmers, system designers and corporate organization managers an intuitive environment for creating and managing system diagrams with separate state transition diagrams (STDs). The STDs are also applicable for system timing.

Model balancing keeps track of system integrity, ensuring that all relevant data flows are utilized in appropriate subtasks, which in SA are modeled with data transformations (DTR). In our case, a DTR represents a C program function, or a set of C functions. With automatic balancing, all inconsistencies caused, for example, by a change in a data flow diagram (DFD) structure, e.g., addition or removal of a data flow, are automatically pointed out or documented. Thus, balancing greatly eases debugging. In our case, data flows model variables, data structures, or bundles of data structures and variables. Model balancing is one of the key features of reliable and efficient software development.

Concurrent documentation management provides automated up-to-date documentation of the system, with a possibility to link created data flow diagrams and textual analysis outputs directly to program documentation.

Program code generation can be done automatically from the SA model, provided that all data flows are appropriately defined in SA model data dictionary, and that at the lowest level of the hierarchical SA representation of the program, textual minispecifications are written strictly as program functions. Also, program timing has to be strictly defined with control structures available in SA.

Program code reusability follows from the well-structured programming and documentation. With SA, it is easy to reuse existing and verified program code in the graphical environment.

As SA provides for hierarchical representation of the program under development, all elements on a given level of hierarchy are presented with the same level of abstraction, and thus *information hiding* is provided for, which clarifies presentation.

Interleaving system development phases, like program specification, programming, and testing, aids especially if the original specification are not strict and exact. For example, it is convenient to show a graphical diagram of a piece of software to a customer for identifying changes and for adding more detail into the specifications as the program development proceeds. With automatic code genera-

0-7803-7005-8/01/$10.00 © 2001 IEEE

tion, a prototype program can easily be assembled at each production stage.

For many systems, object oriented methods, or object oriented computer-aided software tools are often more practical than SA methods, but in general, SA methods are more straight forward, and easier to understand [3].

B. *Structure and Elements of SA Models*

In this work, Prosa/sa of Insoft Oy, Finland, is used. Prosa/sa uses Yourdon structured analysis and design method [1], its data flow diagrams (DFDs) obey Tom DeMarco notation [8] and state transition diagrams (STDs) are done according to Ward & Mellor notation [9]. Bachman and Chen notations are available for data entity relationship diagrams, but these are not needed in this paper.

An SA model is a hierarchical and structured presentation of a system. The three models used in the SA method are illustrated in Fig. 1. *Environment model* consists of a context diagram and an event list. The context diagram contains a single DTR, which represents the whole system, and the outside connections of the system, which are denoted by terminators. Terminators appear only in the context diagram. In the event list, all external events and the corresponding system responses, are listed. The single DTR of the context diagram contains a *behavioral model* of the system, which is a hierarchical system of lower level DFDs, STDs, and textual minispecifications. *Information* model contains data entity relations diagrams.

A DFD of an SA model consists of data transformations (DTRs), control transformations (CTRs), terminators, data and signal flows, and data stores and buffers. These elements are illustrated in Fig. 2, as used by Prosa/sa.

A DTR in a DFD is an abstraction representing a subsystem, and incorporates either a lower level DFD, or at the lowest level of abstraction, each DTR may consist of a minispecification, c.f. Fig. 1. A minispecification is a textual document written in descriptive natural language, or, for example, with a programming language. For example, when modeling a company organization, descriptive language is used in the minispecifications, while in our case, minispecifications are C language functions. Prosa/sa readily supports several header types for minispecification writing, e.g., C, ADA, and free text.

Data and control flows of the environment and behavioral models, c.f. Fig. 1, are presented with the same level of abstraction as DTRs of the DFD. Data flows generally carry data structures or bundles of data. Contents of data flows are defined in data dictionaries (not shown in Fig. 1).

A DFD may contain one control transformation (CTR) if needed. A CTR consists of an state transition diagram (STD). An STD, c.f. Fig. 1, consists of system states, and transitions between the states. Transitions may be associated with conditions for transition, and with actions to be taken when the transition occurs. In Fig. 9, transition conditions are noted above the horizontal lines besides the arrows denoting transitions. Below the same horizontal line, the corresponding actions are listed, which are taken when the transition occurs. The first transition in Fig. 9a is the initial transition, which is taken as the program is started. The second transition in Fig. 9a does not have a condition but only an action associated with it, while the last transition in Fig. 9a carries only a transition condition. In Fig. 9, "T" denotes a trigger signal, which triggers a sequential processing of a DTR.

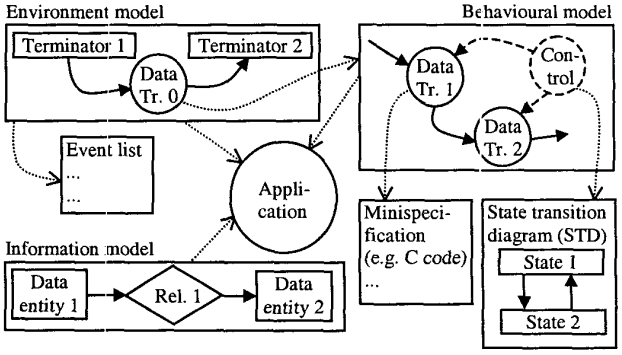

Fig. 1. The three models used in the SA method. The environment model of Netsim is presented in Fig. 3, behavioral model in Figs. 4-8, and STDs in Fig. 9. Adapted from [2].

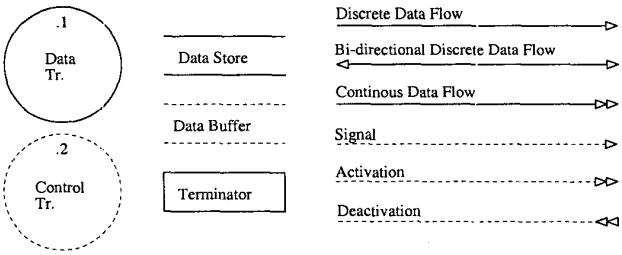

Fig. 2. Elements of structural analysis models as used in Prosa/sa. Adapted from [2].

III. Structured Analysis (SA) Model of Netsim

A. *Netsim*

In this paper, an SA model of Netsim has been created so as to reflect the structure of the original C program, in which a main program calls a number of functions in a given order for a number of times, corresponding to a given simulation time period, e.g. 10 seconds in 1 ms simulation steps. Original C functions correspond to DTRs seen in Figs. 5 – 8. All program variables, e.g., mobile system parameters, communications system state, radio channel behavior, and simulation results, are carried in three C data structures; "Mobile," "Basestation," and "System". The structures "Mobile" and "Basestation" carry parameters of all the mobile units and base stations, respectively. Radio channel behavior is stored in "System" data structure in DTR "***Initialize***," in Fig. 4. The individual C functions are called with appropriate data structures as parameters. In the sequel, the data flow called "Parameter Structures" carries all the three data structures mentioned above. Because of this simple and strictly sequential program data structure, no information model is needed.

Radio channel fading data has been precalculated using ray tracing on a 600 m x 700 m map of Helsinki city center. The pedestrian mobile users are assumed to be walking on a predefined route, which is approximately 1575 meters long. Ray tracing has been calculated from 42087 points of the route to the maximum of seven base stations through five propagation paths. A portion of the users may be defined to be indoor users with higher attenuation. Bandwidth may be set to 5 MHz, and carrier frequency to 2 GHz, for example.

It is to be noted that in this paper, Netsim is described with its original simple algorithms. The more advanced and realistic, algorithms to, or under development with Netsim, e.g. power control and antenna algorithms, fall out of the scope of this paper. Therefore,

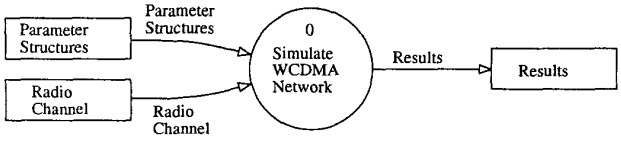

Fig. 3. Environment model of Netsim.

justifications for each presented algorithm must be carefully considered.

B. Environmental Model

The environmental model of Netsim consists only of a context diagram, seen in Fig. 3. In Fig. 3, the terminator "Radio Channel" denotes a data file read, terminator "Parameter Structures" may include both data file read and user specified command line input, and terminator "Results" may denote both data file write and display functions. The single DTR "***Simulate WCDMA Network***," contains the whole simulation program. Since Netsim is a sequentially run simulation program, which does not take any inputs during run time, i.e., it does not observe any events during run time, nor generate any responses to any external events, no event list is needed.

C. Behavioral Model

The lower hierarchy levels of the SA model constitute a behavioral model, in which the system is hierarchically divided into functional elements. The data flow diagram (DFD) of the single Data Transformation (DTR) of the context diagram, Fig. 3, is seen in Fig. 4. DTR "***Initialize***" in Fig. 4, is a C program function used to initialize all the variables, and to read the radio channel behavior data files into the SYSTEM data structure. All other DTRs of DFD seen in Fig. 4 are further defined by lower level DFDs, seen in Figs. 5 – 8.

DFD "***Generate***" in Fig. 4 is seen in more detail in Fig. 5. In Fig. 5, DTR "***Generate MSs***" consists of a minispecification, which generates new mobile speech users if the maximum number of users would still allow more users. Users are generated randomly according to a call birth rate (Poisson distributed with a preset mean), and randomly assigned positions on the route (uniformly distributed on a predefined route), velocities (Gaussian distributed with a preset standard deviation), and call durations (exponentially distributed with a preset mean). Data users are generated analogously in DTR "***Generate Data Users***," Fig. 5. DTR "***Generate MS Traffic***," Fig. 5, contains a minispecification with traffic models with and without voice activity. The traffic model with voice activity uses eight modified eight-state Brady model [10], and transmitter power is set accordingly during periods of speech and silence. In DTR "***Generate MS Data Traffic***," data traffic is generated according to WWW traffic measurements at HUT in 1996. Accordingly, uplink/downlink traffic is heavily asymmetric. Synchronization failures are detected in the traffic generation DTRs according to transmitter power and signal quality, and failed users are removed. With only voice users in the network, and with a call birth rate of 4000-4500 calls/hour, there may be 200-225 users simultaneously active in the network.

A call may go through five stages: call birth, synchronization, admission, conversation/data transmission, and call end. A call may end either in case of no synchronization, no admittance, radio link failure, or through normal call termination.

DTR "***Propagate***" in Fig. 4 contains DTRs "Mobility" and "Propagation," Fig. 6. In the minispecification of DTR "***Mobility***," a new position of each mobile on the route is calculated according to the old position, mobile velocity, and simulation time step. In the minispecification of DTR "***Propagation***", thereafter, amplitudes and phases of the transmission arriving though five propagation paths are calculated by cubic convolution interpolation from the propagation data generated beforehand by ray tracing. "Propagation" is the most computationally demanding DTR of Netsim.

DTR "***Receive***," Fig. 4, is divided in DTRs "Interference," "Interference Downlink," "Averaging," and "Synchronization," Fig. 7. In Fig. 7, DTR "***Interference***" calculates the received power for each uplink path of each link. Each link consists of maximum of five paths. Thereafter, signal-to-interference (SIR) ratio is calculated taking into account maximal ratio combining and processing gain. Analogous SIR calculations are done for downlink in DTR "***Interference Downlink***." DTR "***Averaging***" produces estimates of the average signal strength for handover and admission control functions, which cannot operate on instantaneous estimates of signal strength. For a newly born call, DTR "***Synchronization***" determines synchronization by comparing SIR to a preset threshold. Upon successful synchronization, a call detection time is recorded. If synchronization is unsuccessful, transmission power level is increased in appropriate traffic generation DTR during the next simulation step. Synchronization is deemed unsuccessful, if adequate signal quality is not achieved with maximum transmission power.

DFD "***Control***," c.f. Fig 8, and DTR in Fig. 4, carries the system functions admission control, signal quality estimation, handover control, and power control. In the SA model presented in this paper, simulation time counter is placed in this DFD. In DTR "***Admission***," it is decided, after a preset delay after successful synchronization, whether or not to admit a new call into the system. Admission decision is based on the received signal quality and on the available system resources. DTR "***Quality***" estimates received signal quality, and drops calls on detected quality failure. Here, quality estimation is based on average SIR calculated over a frame duration, and compared to a preset SIR threshold. Quality failure is deemed if SIR stays below a threshold for a preset radio link timeout period. In DTR "***Handover***," handover is decided upon based on average signal strength at base stations. Procedures for both hard and soft handovers are provided. Handover procedure selection is controlled by a preset parameter, which sets the maximum number of base stations that can participate in handover. Set of active base stations participating in handover is maintained based on the average received signal strength, and macrodiversity margin. Marcodiversity margin is the largest allowed difference between the average signal strengths of the strongest and the weakest link in the active set. With the active set full, the weakest link in the active set is replaced by a stronger link that is currently not in the active set, if the average strength of the latter is greater at least by a preset hysteresis parameter. Minispecification of the power control resides within DTR "***Power***." Power control is based on that presented in [11], and considers the instantaneous SIR calculated by "Interference", and employs a single-bit closed loop bang-bang controller with a preset SIR target, maximum and minimum transmitter power settings, and a transmission power change step size, e.g. ±1 dB. Also, a power control loop delay is taken into account.

D. Behavioral Model Control

The Control Transformations (CTRs), seen in Figs 4 through 8, contain the state transition diagrams (STDs) seen in Fig. 9a through e), respectively. The control structure corresponds to that of a main program (DTR "Simulate WCDMA Network," in Fig. 3) calling functions in a strict sequential order. At the end of a simulation step,

simulation time is increased in "Increase Time" DTR in DFD "Control", Figs. 8 and 9e, and next simulation step is started again with "Generate," Fig. 4, and Fig. 9a.

E. Minispecifications

In the case of SA modeling of Netsim, minispecifications are C language functions. Prosa/sa generates the C functions headers automatically from the SA model.

F. Data Dictionary

Components of data flows are defined in data dictionaries. This feature provides for information hiding, and enables presentation of data flows on the same level of abstraction as the corresponding data flow diagram in general. Parts of the data dictionary entry for "Parameter Structures," seen in Figs. 3 – 8, is given in Table I. In Table I, variable type declarations are not shown.

TABLE I
PARTIAL DATA DICTIONARY ENTRY OF "PARAMETER STRUCTURES".

```
ParameterStructures =
Basestation +     * Base station parameter data structure *
Mobile +          * Mobile station parameter data structure *
System            * System parameter data structure *

Basestation =     * C structure containing the following for each BS *
Pt +              * Total transmission power *
bit_rate +        * Transmission bit rate *
id_number +       * Base station identification number*
nr_of_MS          * Amount of MSs connected to *

Mobile =          * C structure containing the following for each MS *
id_number +       * identification number for MS *
data_call +       * identificator of data call *
waiting +         * state indicator for data calls *
call_init_time +  * initialisation time of call *
call_detection_time +    * detection time of call *
call_admission_time +    * admission time of call *
handover_time +          * time of handover event *
call_length +     * generated length for a call *
velocity +        * velocity of the mobile *
location +        * location of the mobile *
    .                    .
    .                    .
    * "Mobile" carries also information on SIR estimates, BERs, *
    * received signal quality, and admission, amongst others *


System =
t +               * Simulation time *
N_ms +            * Number of active voice users *

N_calls +         * Cumulative number of voice users *
N_fails +         * Cumulative number of fails *
N_qual_fail +     * Cumulative number of uplink quality failures *
N_qual_fail_dl +* Cumulative number of downlink quality failures *
N_data_qual_fail_dl +  * Cumulative number of data users *
                       * downlink quality failures *
N_no_adm +        * Cumulative number of admission failures *
N_no_data_adm +        * Cumulative number of data users *
                       * adm failures *
N_no_det +        * Cumulative number of "No detection" -failures *
N_no_ch +         * Failures due to the maximum amount of MSs *
                  * exceeded *
N_no_data_ch +         * Failures due to the maximum amount *
                       * of data users exceeded *
    .                    .
    .                    .
    * "System" carries, for example, all the simulations results, e.g. *
    * counts of approximately ten different types of failures, along *
    * with the basic communications system parameters *
```

IV. CONCLUSIONS

In this paper, a communications system simulator program development project using structured analysis is presented. SA method provides for greater reliability, and maintainability, which may otherwise become obstacles of simulator utilization and development during software lifecycle, especially if the simulator is developed and used by several researchers over several years. Modern SA tools also provide for automated documentation. Though this paper considers a simulator software development, SA method is well usable for various tasks, ranging from company organization and information flow planning, and production plant design, to software component reusability management, and system programming.

Also presented in this paper is the SA model of a WCDMA mobile communications network simulator "Netsim", programmed with C language. The efficient and modular structure of Netsim is to be well presentable as an SA model, which greatly enhances the documentation and maintainability of any software, or other system.

ACKNOWLEDGMENT

The work has been funded by the Technology Development Centre of Finland, Nokia Corp., Finland, Sonera Ltd., Finland, and Radiolinja Ltd., Finland. Institute of Intelligent Power Electronics, HUT, provided the facilities for modeling with Prosa/sa.

First version of Netsim was programmed by P. Bergholm, and a major update was made by A. Pietilä during their time with Communications Laboratory, HUT. Netsim program code was reorganized by J. Hämäläinen, at the time with Laboratory of Signal Processing and Computer Technology, HUT. Also several other researchers have taken part in improving Netsim at HUT.

REFERENCES

[1] E. Yourdon, *Modern Structured Analysis*. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc., 1989.

[2] Insoft Inc., *Prosa/sa, SA Structured Analysis and Design, User's Manual*. Document number IP 952-5173-08-9, Insoft Oy, Oulu, Finland, 2000.

[3] S. Väliviita, P. Tiitinen, and S. J. Ovaska, "Improving the reusability of frequency converter software by using the structured analysis method," in *Proc. IEEE Int. Symp. Industrial Electronics*, Guimarães, Portugal, July 1997, pp. 229–234.

[4] P. Bergholm, M. Honkanen, and S.-G. Häggman, "Simulation of a microcellular DS-CDMA radio network," in *Proc. 1995 Fourth IEEE Int. Conf. on Universal Personal Communications*, Tokyo, Japan, Nov. 1995, pp. 838–842.

[5] A. Pietilä, "CDMA network planning aspects," in *Proc. IEEE XXII Convention on Radio Science and Remote Sensing Symp.*, Otaniemi, Finland, Aug. 1998, pp. 27–28.

[6] A. Pietilä, "Simulations of voice and data traffic in WCDMA network," in *Proc. IEEE 49th Vehicular Technology Conf.*, Houston, TX, USA, May 1999, pp. 2070–2074.

[7] A. Boukalov, S.-G. Häggman, and A. Pietilä, "Study the impact of non-uniform spatial traffic distribution on the system parameters of CDMA Cellular Network," *in Proc. IEEE Int. Conf. on Personal Wireless Communications*, Jaipur, India, Feb. 1999, pp. 394–398.

[8] T. DeMarco, *Structured Analysis and System Specification*. New York, NY, USA: Yourdon, Inc., 1978.

[9] P. Ward and S. Mellor, *Structured Development for Real-Time Systems*. Vols. 1, 2, and 3. New York, NY, USA: Yourdon Press, 1985-86.

[10] H. P. Stern, S. A. Mahmoud, and W. Kin-Kwok, "A model for generating on-off patterns in conversational speech, including short silence gaps and the effects of interaction between parties," *IEEE Trans. Vehicular Technology*, vol. 43, pp. 1094–1100, Nov. 1994.

[11] S. Ariyavisitakul, "SIR-based power control in DS-CDMA systems," in *Proc. IEEE 1992 Global Telecommunications Conf.*, Orlando, FL, USA, Dec. 1992, pp. 868–873.

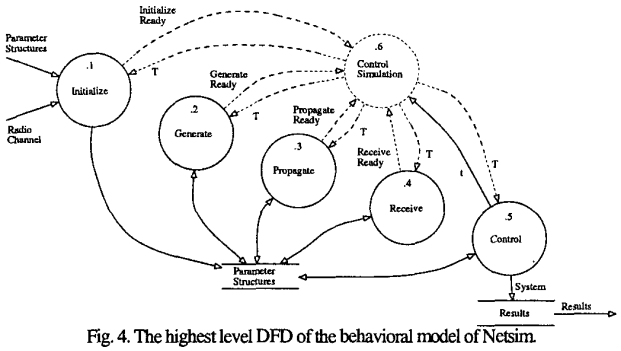

Fig. 4. The highest level DFD of the behavioral model of Netsim.

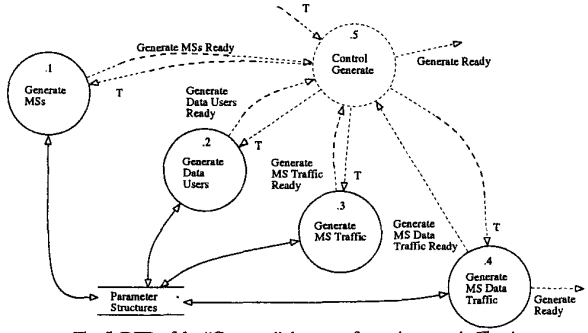

Fig. 5. DFD of the "Generate" data transformation seen in Fig. 4.

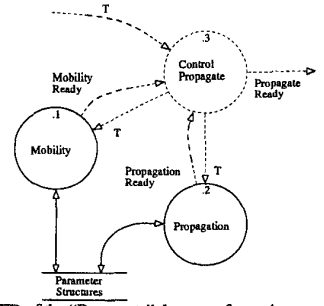

Fig. 6. DFD of the "Propagate" data transformation seen in Fig. 4.

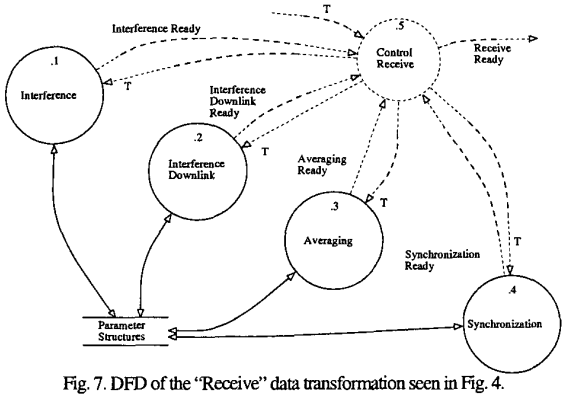

Fig. 7. DFD of the "Receive" data transformation seen in Fig. 4.

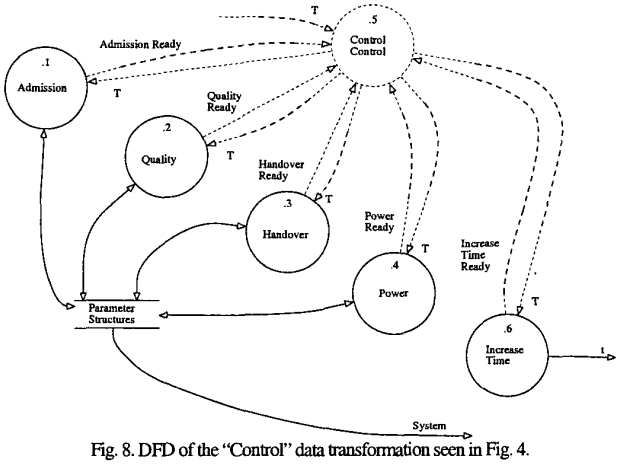

Fig. 8. DFD of the "Control" data transformation seen in Fig. 4.

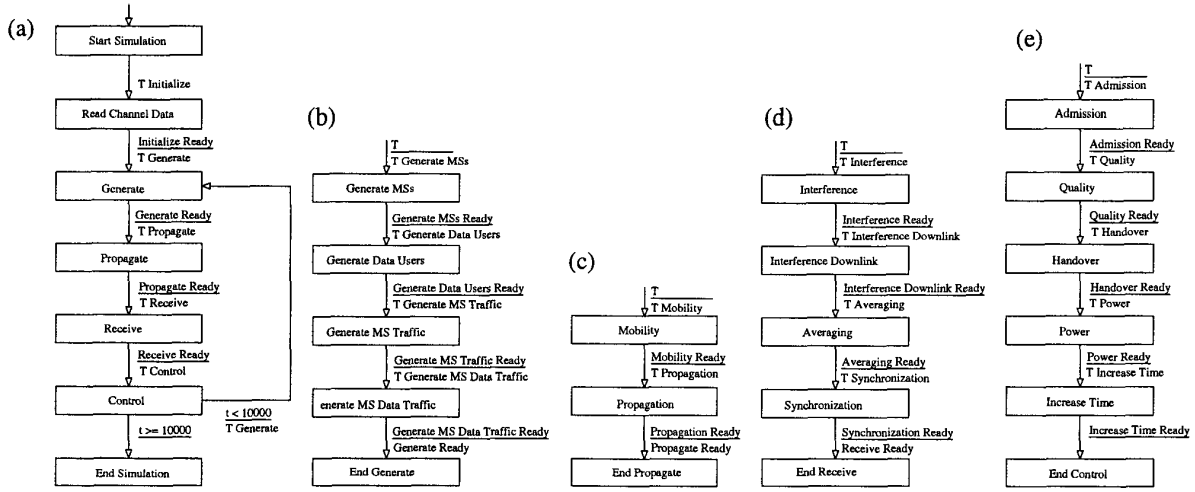

Fig. 9. State transition diagrams of the control transformations; (a) Control Simulation, (b) Control Generate, (c) Control Propagate, (d) Control Receive, and (e) Control Control. T denotes a trigger signal.