Project Number	AC095
Project Title	ASPeCT: Advanced Security for Personal Communications Technologies
Deliverable Type	M (Major)

Deliverable Number	AC095/DOC/RHUL/072/WP22/A
Contractual Date of Delivery	January 1997(Y2M11)
Actual Date of Delivery	January 1997 (Y2M11)
Title of Deliverable	D08: Fraud Management tools: First Prototype.
Contributing Work Packages	WP2.2
Nature of the Deliverable	R (Report)
Editors	Peter Burge and John Shawe-Taylor, RHUL

Abstract	This report demonstrates the use of Rule Based and Neural Network techniques for detecting fraud in mobile telecommunications networks using tools developed within the ASPeCT project. This document discusses the following topics: • A description of the datasets used for the demonstration. • Pre processing Toll Tickets. • Simulating a real-time processing environment. • A common framework for the demonstrations. • Demonstration of the rule based approach to fraud detection. • Demonstration of a neural net based approach to fraud detection using unsupervised learning. • Demonstration of a neural net based approach to fraud detection using supervised learning.

Keywords	ACTS, ASPeCT, fraud detection, demonstration

1 CONTENTS

2 Executive summary

Based on earlier work outlined in deliverable D06 [5] the ASPeCT team have now developed the first prototypes of three fraud detection tools. Two of the tools are based on a Neural Network approach and the other utilises a rule based system. It is the aim of this document to demonstrate that the first prototypes of the tools indeed work and that the results are promising. The fraud detection tools will be demonstrated by applying them to a two months download of Toll Tickets related to new subscribers to the Vodafone Network.

We discuss the pre-processing of Toll Tickets to reduce them to a suitable format for the programmer/fraud detection tool interface. Six fields are taken from each Toll Ticket for use by the first prototypes namely, TT-CHARGED-IMSI, TT-CHARGING-START-DATE,TT-CHARGING-START-TIME, TT-CHARGEABLE-DURATION, TT-NON-CHARGED-PARTY and TT-B-TYPE-OF-NUMBER.

The fraud detection tools themselves have been developed in a modular way such that they will fit within a common framework for the purposes of the demonstration. The framework comprises a number of C, C++ and Perl software simulations to emulate a real time processing environment. Firstly we simulate the billing mediation device to send Toll Tickets out to the fraud detection tool with a pre-set mean interval between each ticket and variance based on a Poisson distribution. These Toll Tickets are also passed to a monitoring tool which checks for alarms being raised by the fraud detection tool. The monitoring tool then stores toll tickets for any subscribers exhibiting suspicious behaviour. The monitoring tool also keeps files containing the current day's and previous day's Toll Ticket in files sorted by IMSI. This is for the purposes of the Audit Trail as advised by WP2.6.

The rule based fraud detection uses the Protocol Data Analysis Tool (PDAT), described in D06, as its central component. PDAT has been considerably adapted from its original purpose of performing audit trail analysis for UNIX systems. PDAT is now able to understand the concepts of user profiles and the format of Toll Tickets. We use a refined Current User Profile(CUP), defined in D06, comprising a number of smaller CUP's that define the medium term past. The start of the day is defined at different times for different subscribers so as to distribute the workload over a 24 hour period. Associated with these CUP's is a life span. User Profile Histories (UPH's) are updated from the CUP's, each time a CUP exceeds its life span, forming a long-term behaviour pattern. Exponential fading is used to maintain these profiles. The differential analysis is performed each time a UPH is updated or when an absolute analysis finds something strange happening. An invoked rule may also request a differential analysis. We utilise the GDBM database for storing user profiles which together has enabled us to reach our performance target.

The Neural Network fraud detection tool, that uses unsupervised learning, develops prototypes of Toll Tickets that span the two dimensional space of call start time verses call duration. Statistical user profiles are then developed by a process of classifying Toll Tickets as prototypes. Unique user profile records are maintained with decay factors and stored in a GNU database for each subscriber. Once a predetermined learning period passes, where the user profiles are first formed, detection begins. CUP's are compared with UPH's by computing the Hellinger distance between them. Alarms

are raised if this value exceeds a currently pre-set threshold value. These alarms then interact with the demonstration framework. The results of using the neural network prototyper are displayed and example profiles of both a subscriber who raised and alarm and one displaying normal usage are shown.

For the Neural Network fraud detection tool, that uses supervised learning, feature extraction plays an important role. Due to the drop in mobile phone usage in the early hours of the morning, we re-parameterize time such that usage is equally distributed. We reduce statistical measures from call durations and intervals between calls to form User Profile Records comprising both short term and long term means and standard deviations for both national and international calls. The basis for the fraud detection engine is a multilayer perceptron. Training is performed using 300 Toll Ticket histories of subscribers whose usage is considered to be normal and 300 histories of subscribers who have had there service terminated through a breach of a velocity trap. The latter Toll Tickets were provided by VOD and were converted from the TACS network. Training consists of an adaptation of weights on connections between the nodes of the multilayer perceptron. These weights are adapted to minimise a suitable cost function using a gradient descent technique known as the Levenberg-Marquardt algorithm. To ensure that training was successful, we further cross-validate with data new to the system.

3 Document Control

3.1 Document History

This is the final version of D08.

3.2 Changes Forecast

(a) Final version of D08 submitted to commission. (29-JAN-97)

3.3 Change Control

In conformance with the project Quality Plan [1].

4 Document Cross References

[1] ASPeCT Quality Plan .

[2] Project Technical Annex (Year 2 Form M4DD).

[3] ACTS AC095, project ASPeCT, “Initial report on security requirements”, AC095/ATEA/W21/DS/P/02/B.

[4] ACTS AC095, project ASPeCT, “Project Trial Plan”, AC095/PFN/W12/DS/P/017/C.

[5] ACTS AC095, project ASPeCT, “Definition of Fraud Detection Concepts”, AC095/KUL/W22/DS/P/06

5 Abbreviations and Glossary of Terms

ASPeCT	Advanced Security for Personal Communications Technologies
AU	Analysing Unit
AUA	Absolute Usage Analyser
CU	Controlling Unit
CUP	Current User Profile
DUA	Differential Usage Analyser
EIR	Equipment Identification Register
GUI	Graphical User Interface
GSM	Global System for Mobile Communications
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
MA	Master Analyser
MSC	Mobile Services Switching Centre
MSISDN	Mobile Station Integrated Services Digital Network
PABX	Private Automatic Branch Exchange
PDAL	Protocol Data Analysis Language
PDAT	Protocol Data Analysis Tool
PSTN	Public Switching Telephone Network
RCF	Roaming Call Forward
SIM	Subscriber Identity Module
SP	Service Provider
TACS	Total Access Communications System
TMN	Telecommunications Management Network
TT	Toll Ticket
UMTS	Universal Mobile Telecommunications System
UPR	User Profile Record
UPH	User Profile History

6 Introduction

This document demonstrates the first prototypes of the three fraud detection tools, developed by work package 2.2 of the ASPeCT project AC095. The documentation for the demonstrators is organised as follows.

- **A description of the data sets used for the demonstration.**

This section describes the nature of the Toll Tickets that will be used in the demonstration including any simulations of fraud scenarios.

- **Pre-processing Toll Tickets.**

This section describes how raw Toll Tickets are pre-processed by extracting parameters used by the fraud detection tools. In addition, sanitised fields are converted such that all characters are visible.

- **Simulating a real-time environment.**

This section describes how we simulated a real-time processing environment by developing perl scripts that simulate the task of the billing mediation device. In addition to this we simulate worm devices for storing the Toll Tickets of users who have raised alarms in the FDT. We also store the past two days worth of Toll Tickets for use in the audit trail.

- **A Common framework for demonstrations.**

In this section we describe how the fraud detection tools fit within the simulation framework outlined in the previous section. We describe the common elements of the framework that the demonstration utilises.

- **Demonstration of the rule based approach to fraud detection.**

This section demonstrates the first prototype of the Siemens rule based fraud detection tool.

- **Demonstration of a neural net based approach to fraud detection using unsupervised learning.**

This section demonstrates the first prototype of the Royal Holloway neural net based fraud detection tool that uses unsupervised learning.

- **Demonstration of a neural net based approach to fraud detection using supervised learning.**

This section demonstrates the first prototype of the KUL neural network based fraud detection tool that uses supervised learning.

7 A description of the data sets used for the demonstration.

For the purposes of the demonstration, input data to the fraud detection tools will be taken from a two months download of GSM Toll Tickets provided by Vodafone. This dataset contains Toll Tickets produced by all the new subscribers to the Vodafone network for this two months. Work retrieving longer Toll Ticket histories, for a selection of users, is still in progress by both Vodafone and Panafon as part of their ongoing contribution to WP2.2. The Neural Network fraud detection tool that uses supervised learning also considers a subset of Toll Tickets that has been converted from the TACS format to GSM. This set of Toll Tickets is for subscribers to the TACS network who have had their service terminated due to a velocity trap check, indicative of cloning.

When considering the two months download of Toll Tickets for new subscribers,the majority will be regular users. The probability that one of our new subscribers, from this small dataset, is a fraudster is exceedingly small. For the purposes of the demonstration the ramaining fraud detection tools have tuned appropriate parameters such that they are more sensitive to changes in user behaviour. We are thus able to label behaviour that is erratic, yet not necessarily fraudulent, as being indicative of fraud. Clearly if we can detect erratic behaviour at this level, any more significant change would be detected too.

In the next section we discuss a filtration process applied to the Toll Tickets prior to storing them in a file for usage by the mediation device simulator.

8 Pre-processing Toll Tickets.

Within our project, the first step was to identify the components which might be relevant for mobile phone fraud. Toll Tickets (TTs) in GSM-networks are defined in the standardised Eurobill Archived Toll Ticket format. During a WP2.2 meeting, 25 components out of the original 68 components have been selected. The number of components a TT actually needs, depends on its type (e.g. Mobile Originating Call: 43 fields, Mobile Originating SMS: 20 fields). To simplify matters, all TTs provided by the network operators consist of the selected 25 components. The field's values are set to

undefined if that component is not supported. These toll tickets are currently used for the development of the fraud detection tools and will be used by the demonstrators.

Since the network operators have to guarantee confidentiality for their subscribers all data fields with references to the user's identity (e.g. IMSI, B-number) have to be sanitised. For this purpose a symmetric ciphering algorithm is used in a 7-bit cipher mode. The fields are enciphered without additional context and thus allow the comparison of these fields between different TTs. Thus, it still can be checked whether two fields refer to the same item (user or destination).

The sanitisation process produces toll tickets in a format, which is not very comfortable to deal with. There are no separators between toll tickets or between different toll ticket fields. For this reason the toll tickets are prepared for further processing. Beautifying the format of the toll tickets is the very first step done within the common FDT-framework. It comprises in detail:

- insert blanks between the TT fields
- insert a carriage return after each TT
- replace each ciphertext character, which is in general a non-printable (e.g. <CR>, Bell, etc.), by two printable, hexadecimal characters.

After this procedure, which is done by a simple C-program, TTs can easily be handled by standard UNIX commands like cat, grep, sort, tail, etc. Shell- and PERL-scripts allow more sophisticated treatment. Changes in such scripts do not require an extra compilation. This way we go along with rapid prototyping for our FDT framework. The "nice" format allows to easily choose the subset of only those toll ticket components which are actually exploited by the fraud detection tool. (e.g. "cut -w 2,3,4,5,6,25 ttfile" produces a file of TTs containing only the most relevant fields 2-6 and 25). Altogether, this more than compensates for the only disadvantage of the "nice" format, which is the increased size of the sanitised TT field.

9 Simulating a real-time environment.

Although the first prototype will not work in a real-time environment, we ultimately want to develop the fraud engine for use in a real-time on-line environment. Moving towards that goal therefore means reproducing the conditions of real-time operation as closely as possible. For this purpose, we have developed a simulation environment consisting of three main modules: the mediation device simulator, the fraud detection tool, and the monitoring tool. We will describe the mediation device simulator and the monitoring tool in this section. We will describe the fraud detection tools in a later section, but will focus here on how they fit within the simulation environment.

Since no real-time mediation device is yet available to the Network Operators, the first step in building the simulation environment is the simulation of the mediation device. The available data consists of a set of files containing all the toll tickets that the switches of the Network produced

over some period of time (see Section 7). For the first demonstrator, we decided to consider only the six most important toll ticket fields: TT_IMSI, TT_CHARGING_START_DATE, TT_CHARGING_START_TIME, TT_CALL_DURATION, TT_NON_CHARGED_PARTY, and TT_B_TYPE_OF_NUMBER. All other fields are ignored at this point. For a given switch, the network records toll tickets according to their order of arrival at the switch. This means that they are roughly ordered according to their TT_CHARGING_START_DATE and TT_CHARGING_START_TIME. However, the activity of the different switches is concatenated, which means that, overall, the toll ticket files do not possess any kind of ordering with respect to time. In a real-time situation, a clock is available for the timing of interesting events. However, in a simulated environment, no clock is available because we may want to execute simulations faster than real time. This lack of an absolute clock is a serious problem for the fraud detection tool. For example, if we want to measure the time interval between two consecutive toll tickets for the same user, it is possible that the date and time of a toll ticket in the data files be posterior to the date and time of the next toll ticket. Therefore, it will be necessary, for all simulations requiring the measurement of time intervals, to sort the toll tickets by TT_CHARGING_START_DATE and TT_CHARGING_START_TIME, and to use this information as the time reference.

Once we have resolved this issue, the files are made available to the simulator of the mediation device. We implemented the simulator as a Perl script. Its first function is to output toll tickets to the fraud detection tool and to the monitoring tool. The user chooses the average time interval between two toll tickets. The simulator then reads toll tickets from the data files and outputs them at random time intervals, according to a Poisson distribution with the mean as chosen by the user. The second function of the simulator is to filter the toll tickets to put all international calls in a common format. The problem is that there are two different cases where a toll ticket results from an international call. The first case is when the user pushes the “+”-button before dialling the number. In this case, the TT_B_TYPE_OF_NUMBER (B-type) is set to “01”. The second case is when the user dials the international prefix, for example “00” for Greece, before dialling the rest of the number. In this case, the B-type remains “00”. To avoid confusion between national calls and international calls using the international prefix, we remove the prefix from the TT_OTHER_PARTY_NUMBER (B-number) and the B-type is set to “01”. We treat the exception of, for example, a Greek user dialling “0030” (the country code of Greece itself) before composing the rest of the number, by removing “0030” from the B-number and letting the B-type at “00”.

The second module in the simulation environment is the fraud detection tool. We will describe the three different approaches (rule-based, unsupervised neural network, supervised neural network) in the coming sections. We only describe here which requirements they must meet in order to fit in the simulation environment. The fraud detection tool receives toll tickets in the six-field format at random time intervals from the mediation device. It then processes them, with the help of a databank of user profiles. If the user raises an alarm, the fraud engine sends the alarm to the monitoring tool, and a copy of this alarm is kept in a logbook as part of the audit trail. The alarm must consist of a string with the IMSI of the suspicious user as its first field. The rest of the string is arbitrary and will contain any information that the fraud engines deem relevant to the analysis of the suspicious behaviour. The requirements on the fraud detection tool are thus minimal.

The last module in the simulation environment is the monitoring tool. Its purpose is to collect relevant information about suspicious behaviours so that an operator can reliably decide whether or

not to discontinue a user's service; and to build a transparent audit trail. We implemented the tool as another Perl script. It receives toll tickets from the mediation device simulator, and - synchronously - the alarms produced by these toll tickets from the fraud detection tool. The monitoring tool has two main functions. First, it maintains a list of all monitored users together with the last alarm they raised. The tool monitors a user from the time it has first raised an alarm until it has not raised any further alarms for two days. A daily list of new monitored users is also kept as part of the audit trail. Secondly, it maintains a list of all calls made by monitored users while they are being monitored. Further, since the fraudulent behaviour may have been going on for a while before it was first detected, all calls made by the monitored users in the two days *before* they first raised an alarm are also added to this list. To accomplish this task, we have to keep two buffers containing all of today's and yesterday's calls. When a user first raises an alarm, all his calls from the last two days are retrieved from these buffers and put in the list of monitored calls. A daily list of monitored calls is also kept as part of the audit trail.

The simulation environment thus provides the following functionalities. It simulates the mediation device in order to operate as closely to real-time on-line operation as possible. It raises alarms for suspicious behaviour encountered by the fraud detection tool; and it collects relevant information about suspicious behaviour in the monitoring tool. The monitoring tool continuously provides the operator, responsible for deciding to discontinue phone service to fraudulent users, with the following information: a list of suspicious users together with their current alarms, and a list of the calls made by these users. It also keeps a logbook of all alarms, all monitored users, and all monitored calls as a basis for the audit trail.

10 A common framework for demonstrations.

Given that the requirements for the integration of the fraud detection tools in the simulation environment are minimal, it is possible to integrate all three prototypes of the fraud detection tool (rule-based, unsupervised neural network, supervised neural network) within a common framework. If all three fraud detection tools are used in parallel, we can view them as a single - more powerful - fraud detection tool combining the strengths of all three prototypes. The three prototypes receive toll tickets from the mediation device simulator simultaneously. They each check the behaviour of the user according to their own criteria. They then may or may not raise an alarm. All these alarms are then passed to the monitoring tool that will manage them within a single framework (the tool will monitor a user if he raises an alarm in one of the fraud engines; and it will maintain a list of monitored users with their alarms and their calls). Another possibility is to use one of the fraud engines as a filter for the others. A simplified version of the fraud engine could detect users having an absolutely *non suspicious* behaviour and reject them from further analysis. This would decrease the load on the other systems and allow them to use techniques that are computationally more expensive. The simulation environment thus proves to be a flexible tool to support the development of the fraud detection tools.

11 Demonstration of the rule based approach to fraud detection.

The subsequent sections describe the rule based approach in more detail. Basic ideas, concepts and the architecture have been explained in D06 and are referred here only. The Protocol Data Analysis Tool (PDAT) has become the central part of the rule based fraud detection tool. Much effort was done to adapt PDAT, whose original purpose was audit trail analysis for UNIX systems, to the new problem of fraud detection. The main tasks where the introduction of user profiles stored in a data base and the realisation of a new protocol that allows PDAT to understand both, user profile as well as toll ticket formats. Once established, PDAT provides a comprehensive infrastructure based on a graphical user interface (GUI) for showing alarms and for editing alarm criteria during runtime.

11.1 User Profiling

The first demonstrators are confined to using the six most fraud relevant toll ticket (TT) components, which are: Charged_IMSI, Charging_Starte_Date, Charging_Starte_Time, Chargeable_Duration, B_Type_of_Number and Non_Charged_Party. Toll tickets are provided by the mediation device via a named pipe. Each incoming toll ticket causes an update of the Current User Profile (CUP) of the respective user.

CUP

The CUP realised in this first demonstrator extends over a 24 hours interval i.e. reflects the user behaviour of a quite short-term past. This time interval may of course be set to a different value. In order to spread the workload for computing a User Profile History Update (UPH, see below)

uniformly this 24 hour interval may start at different times of the day for different users. For one user, the starting time is always the same. According to the concepts in D06 a CUP can be refined by forming a sequence of CUPs $(CUP)_i$, i = 1,..,m to store detailed user information up to the medium-term past. The first prototype realises the sequence (CUP_0, CUP_1) only. CUP_1 is the last but one user profile and is guaranteed to be finished, while CUP_0, the profile of the current day, is in general under construction. During differential analysis the complete sequence will be considered. The fields of a CUP are as follows:

- **cup_start** (starting date and time of the CUP)
- **cup_nr_nat** (total number of national calls during time interval)
- **cup_nat_dur** (total duration of national calls during time interval**)**
- **cup_nr_int** (total number of international calls during time interval)
- **cup_int_dur** (total duration of international calls during time interval**)**
- **cup_warning_type**
- **cup_alarm_type**

On each incoming toll ticket, the fields of CUP_0 are updated.

It is also checked for each incoming toll ticket whether a CUP has completed its life span.

If the time difference between the current time of an incoming toll ticket and ‘cup_start’ is greater or equal the chosen time interval (here 24 hours), but less than twice that interval (48 hours) the following steps are performed:

– update UPH with the sequence (CUP_0, CUP_1) (as described below)

– CUP_1 := CUP_0**;**

– compute CUP_0.cup_start ($date_0 = date_1+1$)

– set all other fields of CUP_0 **to values of incoming toll ticket;**

If the time difference between the current time of an incoming toll ticket and ‘cup_start’ is between n*interval and (n+1)* interval for n>1 then the above procedure is carried out n times, thereby introducing “intermediate“ CUPs with values set to 0 (except for cup_start) for the intermediate intervals with no activity.

Suspicious events are classified into the three categories ‘message’, ‘warning’ and ‘alarm’. Messages might by useful information for the administrator but are not stored in the user profiles. A warning might in combination with an other warning cause an subsequent alarm. The fields ‘cup_warning_type’ and ‘cup_alarm_type’ are used like bit-vectors to store, which kind of warning or alarm has been raised during the time interval. These components prevent the absolute analysis from creating more than one alarm, once a threshold is exceeded.

UPH

The UPH, which represents the long-term behaviour of a subscriber, consists of the following fields:

- **uph_date_time** (for recovery purposes)
- **uph_nr_nat** (long-term average number of national calls during time interval)
- **uph_nr2_nat** (long-term average squared number of national calls during time interval**)**
- **uph_nat_dur** (long-term average duration of national calls during time interval**)**
- **uph_nat_dur2** (long-term average squared duration of national calls during time interval**)**
- **uph_nr_int** (long-term average number of international calls during time interval)
- **uph_nr2_int** (long-term average squared number of international calls during time interval**)**
- **uph_int_dur** (long-term average duration of international calls during time interval**)**
- **uph_int_dur2** (long-term average squared duration of international calls during time interval**)**
- **fraud_probability**

The UPH will be updated whenever a CUP has completed its live span This is done by decaying the current UPH-values and adding "fresh" data taken from the existing CUPs. We use exponential fading to compute the components of the UPH:

For the components UPH.x with x = uph_nr_nat, uph_nat_dur, uph_nr_int or uph_int_dur we set

UPH_{new} .x = (1-f_m) UPH_{old} .x + $\Sigma_{i=0,..,m-1} f_i \, CUP_i$ .y ,

with m=2, and fading-factors f_i, $0 \le f_i < 1$, $\Sigma_{i=0,..,m} f_i = 1$ and with y = cup_nr_nat, cup_nat_dur, cup_nr_int or cup_int_dur.

For the components UPH.x with x = uph_nr2_nat, uph_nat_dur2, uph_nr2_int or uph_int_dur2 we set

UPH_{new} .x = (1-f_m) UPH_{old} .x + $\Sigma_{i=0,..,m-1} f_i \, CUP_i^2$ .y ,

with m=2, and fading-factors f_i, $0 \le f_i < 1$, $\Sigma_{i=0,..,m} f_i = 1$ and with and with y = cup_nr_nat, cup_nat_dur, cup_nr_int or cup_int_dur.

Besides, the fraud-probability is re-computed out of the previous value plus the warnings and alarms stored in the CUP. Since the long-term behaviour is not dependent on a single toll ticket, the UPH is not updated on each incoming toll ticket. The update is still triggered by a toll ticket, but not before the CUP's time interval (24 h) is finished. Depending on the arrival of the triggering toll ticket, this may result in a long time between two UPH updates, however, without causing a delay in the recognition of fraud.

Differential Analysis

The differential analysis is invoked on several occasions:

1.) during each UPH-update

2.) if the absolute analysis found something strange, which caused a message at least.

3.) if any rule dictates an explicit differential analysis

Part of the differential analysis is comparing the fields of the CUP and the UPH in pairs and check the difference against several thresholds. These thresholds will be expressed using the mean *M* and standard deviation σ of the entries of the CUPs, i.e. of the user's day-to-day behaviour.

They can be estimated by the entries of UPH as follows:

set M = **uph_nr_nat,** set σ^2 = **uph_nr2_nat - (uph_nr_nat)^2 .**

(similarly for the other entries of UPH).

The GDBM data base, which has proven to be a very fast data base and the restriction to six TT fields guarantee for fulfilling the network operator's performance needs. If future requirements ask for even more performance, this can be achieved by realising a swapping technique, as specified in D06. Note also, that the fraud detection problem can by its nature easily be parallelised.

11.2 PDAT Infrastructure

PDAL language concepts

Important goals were flexibility and broad applicability, including the analysis of general protocol data, which is achieved by the special language PDAL (Protocol Data Analysis Language). PDAL allows the programming of analysis criteria as well as a GUI-aided configuration of the analysis at runtime. The language's concept is related to AWK, but is enhanced with especially the following:

- A **protocol format** describes the possible content of the several data records. The supported data types are int, float, string and time-stamp. The current protocol format describing toll tickets and user ticket allows PDAL to refer to the values as base for the following features.
- An **analysis criterion** formulates a condition on single data records or on a pattern of several data records. There are three types of analysis criteria (in growing complexity): filter, criterion, and behaviour.
- A **rule** assigns a specific action to an analysis criterion. If the criterion is fulfilled, this action will be performed. Typical actions are warnings or alarms shown at the GUI or a certain system call.
- An **analysis configuration** consists of a list of rules.

Except the protocol format of course, all of these concept can be changed during runtime. The threshold of a filter, or a tree of filters a criterion consists of, or rules may be added to or deleted from a analysis configuration.

Dynamic tables are a means for recognising patterns of information across several data records. They allow storing intermediate results in associative arrays, which can be indexed by every data type (string, int, float, ...).

GUI features

PDAT is designed in a manager/agent-architecture, with the Controlling Unit (CU) as manager and the Analysing Unit (AU) playing the agent's role. The essential part of the Controlling Unit (CU) is a GUI, that supports all the administration tasks, which are configuration of analysis criteria and rules, controlling the

AU and displaying all warnings and alarms received from the AU.

Figure 11-1 shows the GUI of PDAT during operation. The PDAT Desktop is the root window of the GUI, which shows a menu list at the top, an alarm display, and additional status information and statistics at the right side. The menu list provides the following features:

PDAT	Control of the CU
Host	Control of the selected AU. In general we can deal with several AUs, possible operations are: connect, disconnect, transfer (of a change configuration) and kill.
Report	Detailed description of alarms concerning one or all criteria
Configuration	Edit an analysis configuration (set of rules): enable/disable certain rules, load/save set of rule
Critbase	Edit the criterion base (analysis criteria): edit certain criterion, load/save set of criteria

The status information shows which configuration is currently active ('up': user profiles) and to which AU represented by its host we are connected. Statistical information consists of how many user records have been processed and how many events have been transferred to the CU. All alarms and warnings are listed in the main display part. Additionally, the PDAT icon represents the severity of the current event by its colour. The icon looks grey at the beginning, changes to yellow for a warning or to red for an alarm and goes back to grey after clicking the icon for acknowledgement.

The 'Configuration Rules' window gives an impression of how rules are dealt with. Firstly, a new rule can be introduced by using the GUI as an editor. By clicking the 'Check' button a PDAL syntax check is applied to the new rule. Secondly, we can simply change a criterion (e.g. change a threshold) as well as the concerning rule (e.g. change the issued action). Finally, rules can be added to or deleted from the current set of applied criteria.

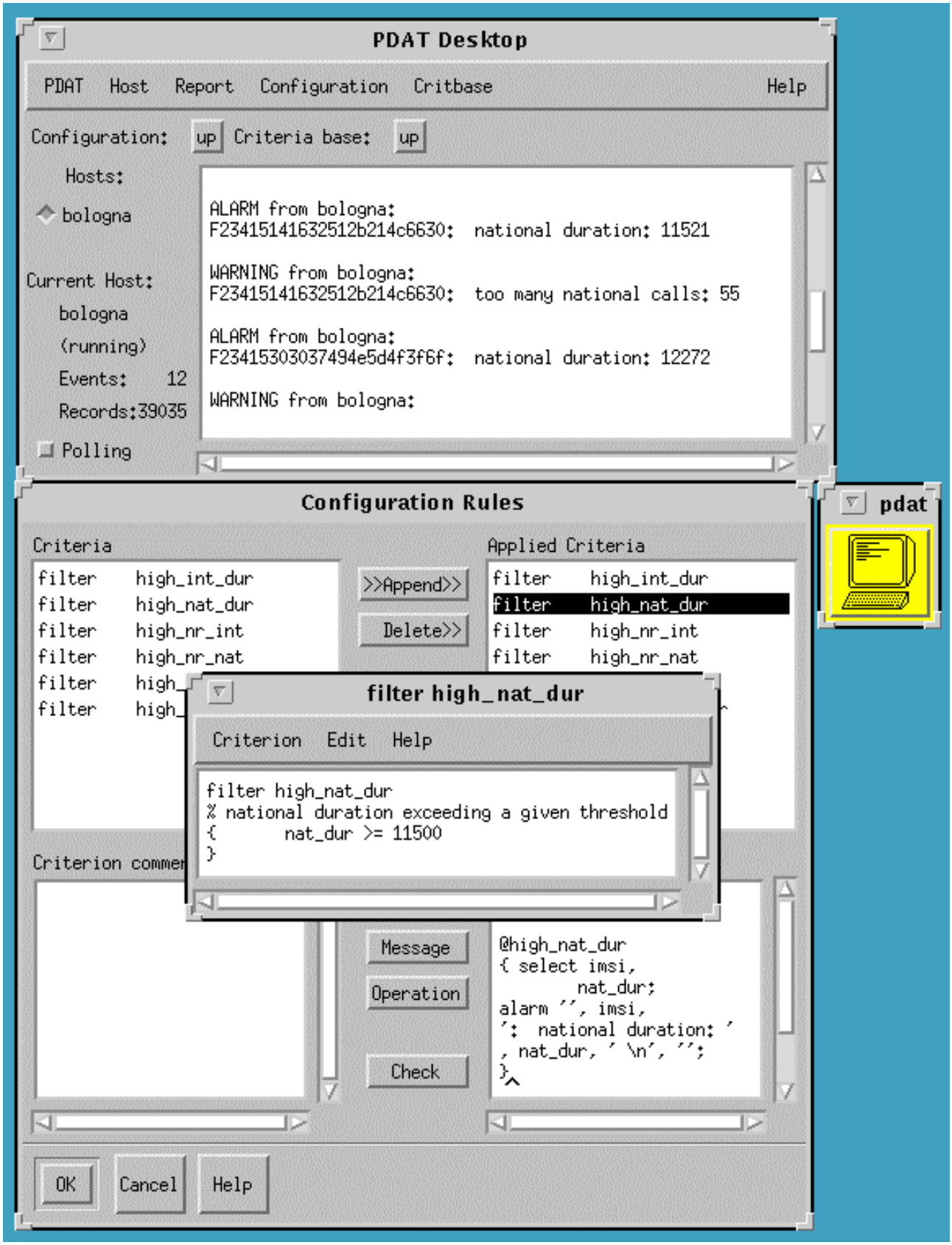


Figure 11-1 Graphical User Interface of Rule Based Tool

12 Demonstration of a neural net based approach to fraud detection using unsupervised learning.

The purpose of this section is to describe and demonstrate the implementation of the first prototype of a Neural Network based fraud detection tool. This particular tool uses unsupervised learning to classify Toll Tickets enabling the construction of statistical user profiles for analysis by a fraud engine. The fraud detection tool sits within the common framework described in section 10.

The first stage in the procedure is developing a set of prototypes to represent a large dataset of Toll Tickets. This task is performed offline by the Neural Network prototyper. The technique is based on the work of Grabec who introduced the Second Maximal Entropy Principle and demonstrated its use in prototyping a one dimensional continuous random variable. The theory for this procedure was presented in D06[5] and has been extended to the multidimensional case for prototyping Toll Tickets.

One of the *tuneable* parameters of the fraud detection tool that needs to be considered at this stage is the number of prototypes *K* that will be used to generate both the Current User Profile(CUP) and the User Profile History(UPH). This first demonstration of the fraud detection tool considers four parameters from each Toll Ticket, namely TT-CHARGED-IMSI, TT-CHARGING-START-TIME, TT-CHARGEABLE-DURATION and TT-B-TYPE-OF-NUMBER. Prototypes of the two dimensional space of start time verses call duration are produced for National calls, International calls and *other* calls, such as calls to voice mailboxes and call forwarding. For the purposes of the demonstration we have chosen 50 prototypes of national calls, 50 prototypes of international calls and 10 prototypes of the remaining *other* calls. Figure 12.1 below shows the process of developing prototypes from Toll Tickets.

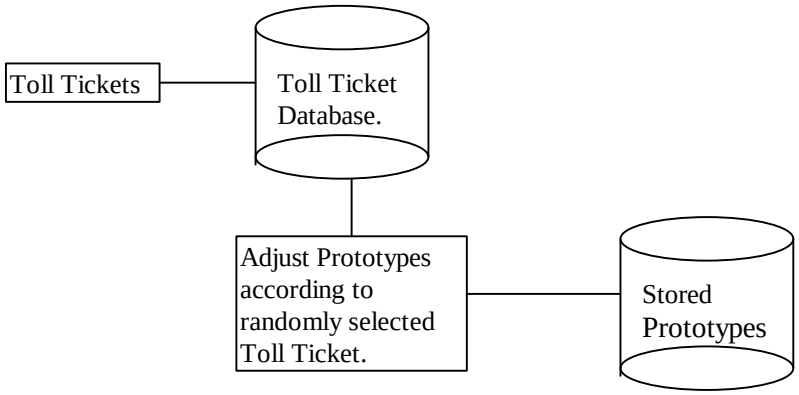
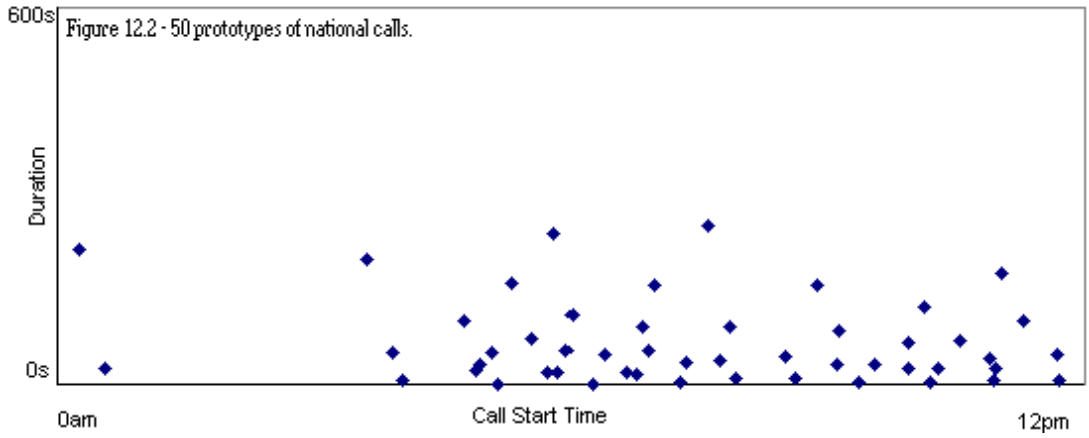


Figure 12.1- Developing Prototypes.

In figures 12.2 and 12.4 we show the results of applying the Neural Network prototyper, using the demonstration criteria given above, to one weeks worth of Toll Ticket data. 50,000 data points of the original Toll Ticket data have also been plotted in figures 12.3 and 12.5 to demonstrate how the prototypes span the real data set. The prototyping technique generates prototypes such that if the real data was classified to its nearest prototype, each of the prototypes would have approximately the same number of classifications..



Both figures 12.2 and 12.4 display 50 prototypes. Figure12.2 is for national calls. Notice that the average call duration is shorter for national calls compared to the international calls seen in figure 12.3. Another point to note is the reduced activity of calls during the early hours of the morning. This indicates that we should be able to perform an analysis on a reduced submanifold of the complete behaviour space, spanned by the full set of Toll Tickets, without significantly reducing the accuracy of the fraud detection tool. This will have a beneficial impact on performance.

Figure 12.3 - 50,000 National Calls Input To Neural Network Prototyper.

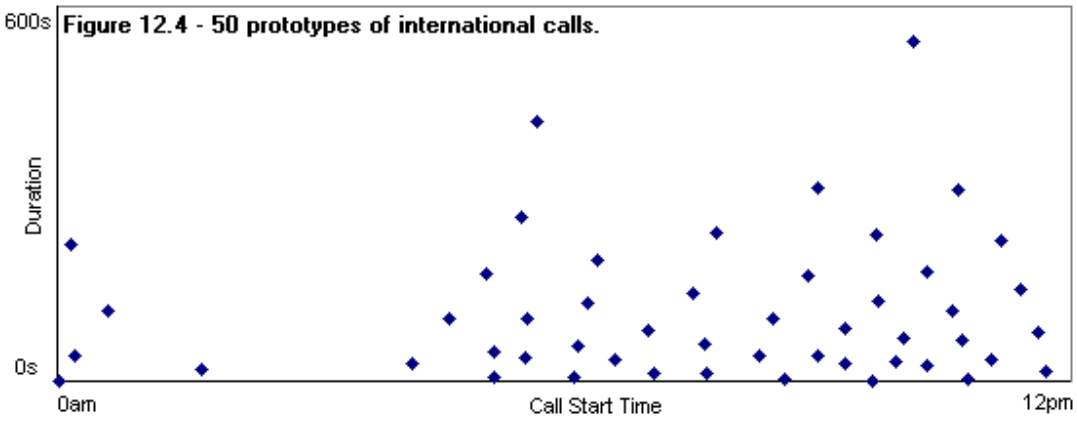
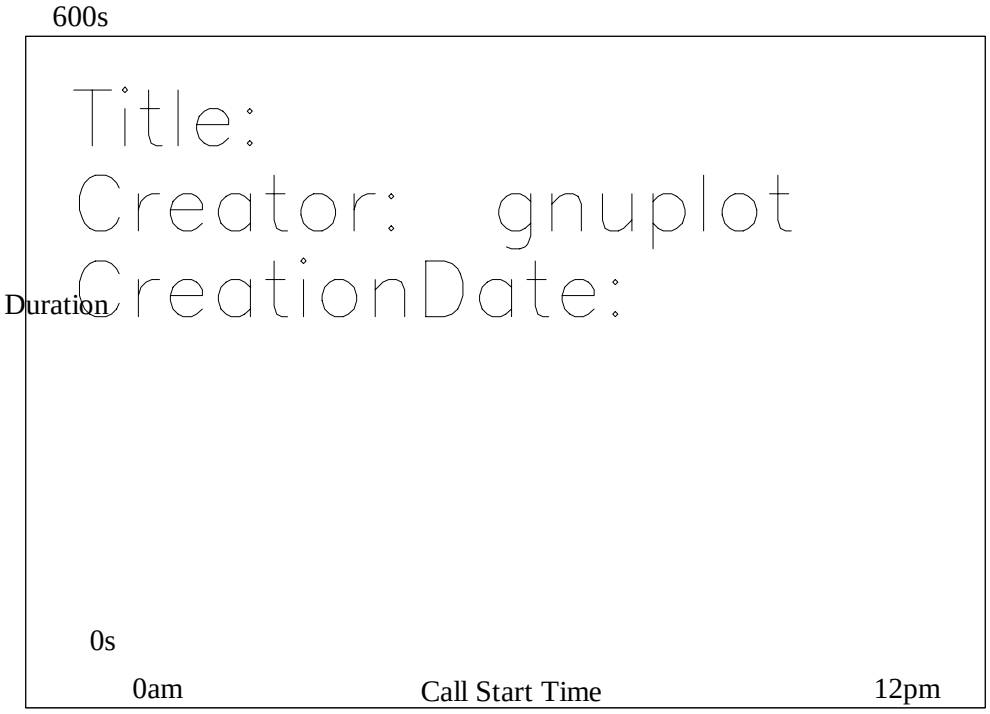


Figure 12.5 - 50,000 International Calls Input To Neural Network Prototyper.



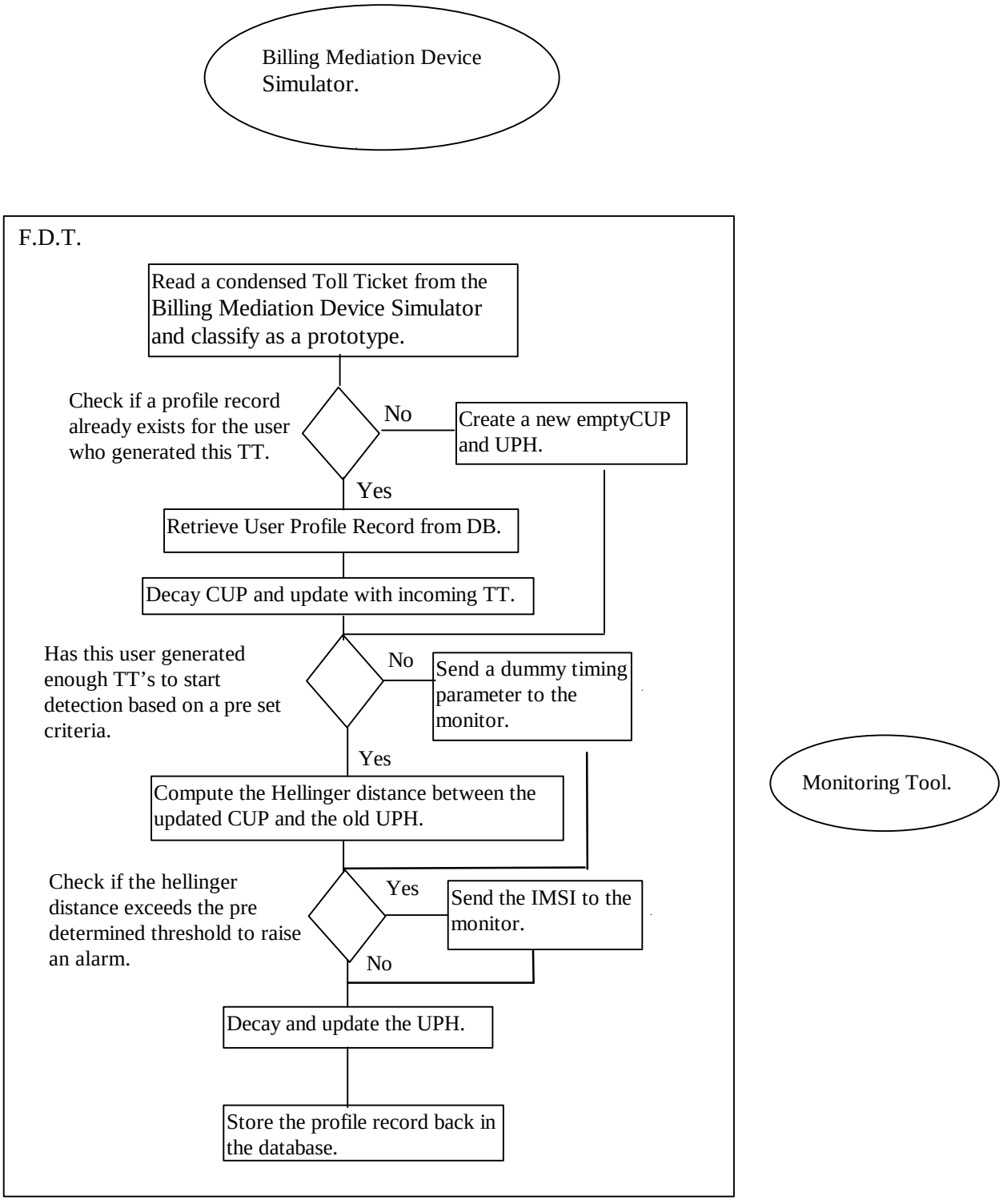
When we start the fraud detection tool within the common simulation framework, the first task of the fraud detection tool is to develop statistical user profiles from the incoming data. A complete user profile record consists of a CUP, a UPH and a counter for the number of Toll Tickets that have been processed for the user under consideration. Figure 6 depicts such a profile record.

110 Prototypes forming the CUP National - International - Other	110 Prototypes forming the UPH National - International - Other	Number of hits.

Figure 12.6 - A statistical user profile record.

A user profile record is unique for each subscriber and forms the entry that is written to, and read from, the database each time a Toll Ticket is processed. Accessing the database of user profile records is the limiting factor on the performance of the system. In practice, for this demonstration, we have still surpassed our real-time processing goal of 30 Toll Tickets per second.

The next figure demonstrates the way in which the fraud detection tool operates within the common framework.



As mentioned previously there are a number of parameters associated with the fraud detection tool that determines the way in which it behaves. Owing to the current lack of fraudulent examples, due to the apparrent inability of the fraudster to successfully and significantly attack GSM technology, we tune the performance parameters of our system to detect erratic usage of a mobile telephone. If the fraud detection tool can detect such subtle changes in behaviour that may still be deemed to be the extremes of normal usage, then clearly this tool would also trap much greater differences in behaviour patterns that is the basis for detection using a differential analysis.

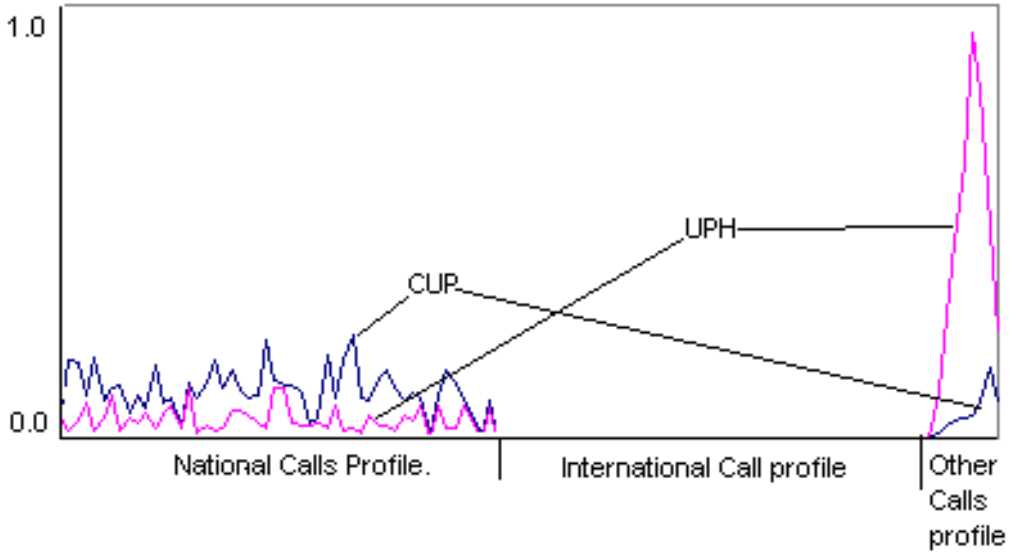
Our experiments therefore concentrate on tuning parameters to see how many users are labelled as having suspicious behaviour and when these alarms were raised. Using the two months download of Toll Tickets for new subscribers, taken from the Vodafone network, described in section 7 we see what effect changing parameters such as

- The required number of Toll Tickets, for each user, that must be processed before a differential analysis is made between the CUP and the UPH.
- The threshold based on the Hellinger distance to determine when an alarm is raised.
- The relative decay factors of the CUP and the UPH.

The analysis of the results and statements on the performance of the system will be made in a later document however some intuitive features are demonstrable at this point.

Figure 12.7 shows the CUP and UPH of a user who raised an alarm at an early stage due to a low Hellinger threshold and a low number of Toll Tickets that were required to start detection.

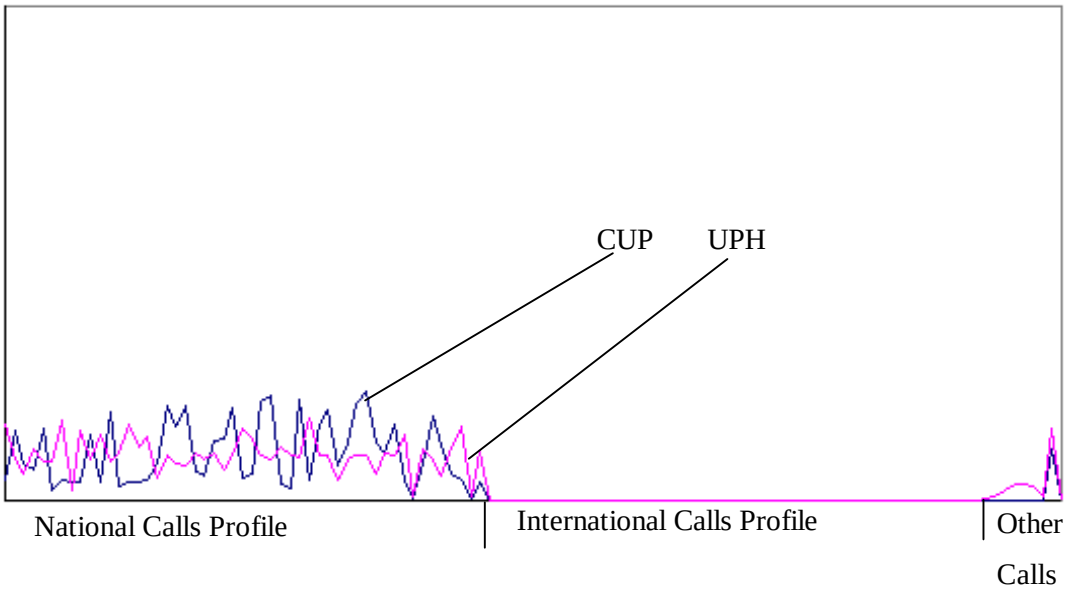
Figure 12.7 - User profile of a subscriber who raised an alarm.



In this particular case the alarm was predominantly raised by a drop in the number of ‘other’ calls. Although not indicative of fraud, it is interesting to note that this users behaviour changed only in the sense of making less ‘other’ calls. With current absolute usage systems only an increase in activity would trigger an alarm. If a subscriber’s phone was stolen, it is unlikely that the thief would check the subscribers voice mail. This system would clearly detect such an anomily and warn the network operator.

In contrast, Figure 12.8 below displays a user profile record for a subscriber displaying what is considered as acceptable behaviour by the demonstration fraud detection tool.

Figure 12.8 - User profile of a subscriber during acceptable usage.



In summary, we have described the implementation of the Neural Network based fraud detection tool that uses unsupervised learning. We have described how the system needs to be performance tuned and have demonstrated what the fraud detection tool considered as normal or suspect usage in two specific examples, based on a differential behaviour analysis. There is much refinement required to optimize the fraud detection tools ability to detect frauds.This will result from the analysis of the system to be performed in the next activity. Initial indications are that the tool works and indeed will prove useful on its own or possibly pipelined with the other two tools being developed, as a filter for acceptable usage.

13 Demonstration of a neural net based approach to fraud detection using supervised learning

The general architecture for the neural network approach to fraud detection is described in Deliverable 06; and we refer the reader to this document for a first presentation of the fraud detection tool and of the real-time environment. We describe here the specifics of the implementation of the fraud detection tool. That is, how this tool extracts the User Profile Record, the Current User Profile, and the User Profile History from sequences of toll tickets; how it extracts the relevant features from these profiles; and how the neural network processes these features to produce its decision.

13.1 Profiling

13.1.1 User Profile Record (UPR)

The six fields that the mediation device simulator provides to the system are the TT_IMSI, TT_CHARGING_START_DATE, TT_CHARGING_START_TIME, TT_NON_CHARGED_PARTY, TT_B_TYPE_OF_NUMBER. This information is stored in the User Profile Record. However, we obtain the Current User Profile and User Profile History by using a filtering technique on the User Profile Record, we therefore need to translate the TT_CHARGING_START_DATE and TT_CHARGING_START_TIME to numerical values. The mediation device simulator converts thus the TT_CHARGING_START_DATE to the number of days from some reference date using a calendar and the TT_CHARGING_START_TIME to the number of seconds from midnight. This way, the absolute time of beginning of the call is equal to (in seconds from the reference date on midnight) TT_CHARGING_START_DATE * 86400 + TT_CHARGING_START_TIME.

13.1.2 Current User Profile (CUP)

The Current User Profile contains the information about the short-term behavior of the user. We have decided to focus on the following measures of the behavior of the user: the duration of calls and the interval between calls. High call duration, and low call interval (high call frequency) are indicators of, for example, call selling; while low call duration and low call interval are indicators of a PABX attack. We further split the call duration and the call interval between national and international calls. We do not take other types of calls into account for the first prototype because no such call is present in the fraudulent data. Furthermore, we do not only compute the mean duration of calls and the mean interval between calls, but also the variance of the call duration and of the call interval.

If we work with the absolute time of a call, we have the problem that calls tend to have very high intervals during the night, and low intervals during peak hours. To compensate for this, we determine

the daily distribution of the calls and re-parameterize time so that the activity is equally distributed. The result is that the "time" difference between 10 pm and 6 am, maybe equal to the difference between 10 am and 10:30 am, because the amount of activity during the two periods is equal. The absolute time of last national call t_n, of last international call t_i, of last other call t_o have to be part of the profile to compute time differences. The short-term averages are computed using a first order filter. At each toll ticket t, we can compute the average $<x(t)>_a$ of a quantity $x(t)$ over all previous toll tickets as follows:

$$< x(t) > a = \begin{cases} (1-a) < x(t-1) > a + a.x(t) & \text{if } x(t) \text{ is defined} \\ < x(t-1) > a & \text{if } x(t) \text{ is not defined} \end{cases}$$

It is important to note that the quantity $x(t)$ might not be defined at every toll ticket; for example, only one of duration of national call d_n, duration of international call d_i, and duration of other call d_o will be defined at a time. The filter is, in fact, a first-order low-pass filter, and it gives thus an estimate of the mean $E(x)$ of the signal $x(t)$ (if the signal is a sequence of independently identically distributed random variables). Further, the filter can track changes in the mean. So, the short-term average of the duration of national calls will be $<d_n>_a$ in our notation. To compute a short-term standard deviation of a quantity $x(t)$, we can still use similar filters but on $x(t)^2$. We derive this from the definition of the variance as follows ($\hat{m}, \hat{s}$ being estimates of the mean μ and standard deviation σ):

$$m = E(x)$$
$$s^2 = E((x-m)^2) \overset{i.i.d.}{=} E(x^2) - (E(x))^2$$
$$\hat{m} = < x >_a$$
$$\hat{s} = \sqrt{\max(< x^2 >_a - (< x >_a)^2, 0)}$$

This finally gives the following 10 fields for the Current User Profile.

- Absolute time of last national call t_n
- Absolute time of last international call t_i
- Short-term average of the duration of national calls $<d_n>_a$
- Short-term average of the duration of international calls $<d_i>_a$
- Short-term average of the squared duration of national calls $<(d_n)^2>_a$
- Short-term average of the squared duration of international calls $<(d_i)^2>_a$
- Short-term average of the call interval between national calls $<i_n>_a$
- Short-term average of the call interval between international calls $<i_i>_a$
- Short-term average of the squared call interval between national calls $<(i_n)^2>_a$

- Short-term average of the squared call interval between international calls $<(i_i)^2>_a$

13.1.3 User Profile History (UPH)

We derive the User Profile History in a similar fashion by filtering the Current User Profile with a first-order filter with parameter β. This means that the User Profile History is, in fact, a second-order filter of the signals. It thus estimates average quantities, but on a longer time scale than the Current User Profile. Processing the call duration and call interval through a first-order filter to obtain the Current User Profile, and again through another first-order filter to obtain the User Profile History minimizes the memory requirements for the updates of the filter, and therefore minimizes the load on the database of user profiles. Furthermore, the difference between the User Profile History and the Current User Profile can be interpreted as a second-order band-pass filter on call duration and call interval. This means that it is affected neither by very short-term variations (let us say, between one call and the next) neither by very long-term variations (therefore allowing us to track long-term changes in the behavior of the user). The difference between the User Profile History and the Current User Profile allows us to detect deviations from the normal behavior of a user. The date of first call is also kept in the profile to determine at which point differential analysis becomes applicable (since, at the beginning, the User Profile History does not contain any relevant information). These considerations results in the following User Profile History (using the same notation as in the previous paragraph).

- Date of first call
- Long-term average duration of national calls $<<d_n>_a>_b$
- Long-term average duration of international calls $<<d_i>_a>_b$
- Long-term average squared duration of national calls $<<(d_n)^2>_a>_b$
- Long-term average squared duration of international calls $<<(d_i)^2>_a>_b$
- Long-term average call interval between national calls $<<i_n>_a>_b$
- Long-term average call interval between international calls $<<i_i>_a>_b$
- Long-term average squared call interval between national calls $<<(i_n)^2>_a>_b$
- Long-term average squared call interval between international calls $<<(i_i)^2>_a>_b$

13.1.4 Feature extraction

The features used by the classifier will not be the content of the Current User Profile and User Profile History directly, but estimates of means and standard deviations. The means are obtained directly at the output of the filters, but the standard deviations must be computed as $\hat{s} = \sqrt{\max(< x^2 > -(< x >)^2, 0)}$, where the bracket denotes a first-order or a second-order filter. This results in the following vector of features, which is the input to the classifier.

- Number of days since first activity
- Short-term mean of the duration of national calls
- Short-term mean of the duration of international calls
- Short-term standard deviation of the duration of national calls
- Short-term standard deviation of the duration of international calls
- Short-term mean of the call interval between national calls
- Short-term mean of the call interval between international calls
- Short-term standard deviation of the call interval between national calls
- Short-term standard deviation of the call interval between international calls
- Long-term mean of the duration of national calls
- Long-term mean of the duration of international calls
- Long-term standard deviation of the duration of national calls
- Long-term standard deviation of the duration of international calls
- Long-term mean of the call interval between national calls
- Long-term mean of the call interval between international calls
- Long-term standard deviation of the call interval between national calls
- Long-term standard deviation of the call interval between international calls

13.1.5 Storing user profiles

User profiles must be swapped between disk and main memory each time a new toll ticket arrives at the fraud detection tool. The performance requirements are severe as peak performance must exceed 30 toll tickets per second. We obtain such performance by using a simple, but optimized database tool called GDBM. The database is accessed by a key, which is the IMSI of the user and provides a content, which is the concatenation of the Current User Profile and User Profile History.

13.2 Supervised Learning

After designing the front-end, we must design the classifier. The front-end processes the toll tickets to produce sequences of user profiles; and then extracts the features needed by the classifier from these profiles. The classifier then maps a vector of features to an alarm value between 0 and 1 using a multilayer perceptron.

13.2.1 Multilayer perceptron

The neural network used in the fraud detection engine is a multilayer perceptron. It is defined as follows. The network is composed of elementary units called neurons. Each neuron produces at its

output a simple nonlinear transformation of its inputs depending on the value of the weights of the network:

$$y = s(\sum_{i=1}^{n} w_i x_i + w_0), \quad \text{where } s(z) = \tanh(z) \text{ or } s(z) = \frac{1}{1+e^{-z}}$$

The neurons are then arranged in a two-hidden-layer network with D inputs, H_1 hidden neurons in the first layer, H_2 hidden neurons in the second layer, and C outputs. The outputs z_m of the network can then be defined as

$$h_{1_k} = s(\sum_{l=1}^{D} w_{kl} x_l + w_{k0})$$
$$h_{2_l} = s(\sum_{k=1}^{H_1} v_{ik} h_{1_k} + v_{i0})$$
$$z_m = s(\sum_{l=1}^{H_2} u_{lm} h_{2_l} + u_{l0})$$

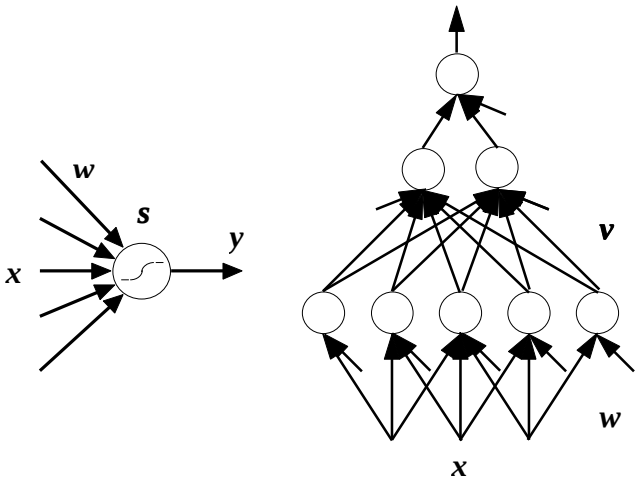


Figure 13.1 Sigmoidal neuron and multilayer perceptron architecture

The main property of multilayer perceptrons is that they can approximate any function of the input to an arbitrary degree of accuracy, provided that enough hidden neurons are available. They can achieve this approximation with a relatively small number of parameters.

13.2.2 Labelling

For supervised learning, we organize the data available for design in a data set of labeled pairs $D = \{(X_1,Y_1),...,(X_K,Y_K)\}$, where Y_k is the fraud label ($Y_k = 0$ for normal behavior, $Y_k = 1$ for fraud) associated to the k-th pattern with features X_k extracted from the user profile. The training data

consists for the first part of the calls made by 300 users from the two-month download from Vodafone; these users are deemed normal. It also consists for a second part of the calls made by 300 fraudulent users. For all 600 users, all the available toll tickets are processed through the front-end of the system to produce sequences of user profiles. We label the sequences of user profiles for the normal users as non-fraudulent. For the fraudsters, we studied the evolution of the profiles over time to determine the beginning of the fraudulent behavior; and we labeled the profiles as fraudulent during the fraudulent behavior and as non-fraudulent otherwise.

13.2.3 Training

The first step is to choose the architecture of the neural network, that is the number of layers, and the number of neurons in each layer. Once we have chosen the architecture, the output of the network is a function of its input X_k and of the parameters w (the weights) of the neural network. There is a discrepancy between the output of the classifier $z(X_k,w)$ and the desired output Y_k. The learning of training of the classifier consists in adapting the weights so as to minimize this discrepancy. The measure of discrepancy is quadratic.

$$\text{find } w \text{ that minimizes } \mathrm{E} = \sum_{k=1}^{\mathrm{K}} \left\| \mathrm{Y_k} - z(X_k, w) \right\|^2$$

We achieve this minimization using a gradient-descent method, namely the Levenberg-Marquardt algorithm. This powerful method is based on algebraic procedures, which permits, given a value of the weights, to determine what modification of the weights would lead to an optimal reduction of the error. After we have modified the weights, we have a smaller error and a new value of the weights. A new correction to the weights is evaluated, so as to reduce the error as rapidly as possible. We repeat this procedure until the error stops to decrease.

13.2.4 Cross-validation

In fact, we split the data set in three subsets: the training set, the validation set, and the test set. In order to maximize the performance on previously unseen data, we use the following procedure called cross-validation. The weights are adapted by minimizing the error on the training set, but we observe the error on the validation set during this process; and we stop the minimization when the error on the validation set reaches a minimum. We then estimate the expected performance on new data by computing the error on the test set.

13.2.5 Determination of the optimal architecture

We determine the optimal weights using the error minimisation procedure, but we have to repeat the procedure to search for a global optimum of the optimisation procedure, since gradient-descent methods are only guaranteed to converge to a local optimum. Furthermore, we have to repeat this procedures for different architectures of the neural network to determine the optimal one. Once we have found the optimal neural network, we simply have to use it on top of the front-end and it will produce an alarm value between 0 and 1 each time a toll ticket is presented to the fraud detection tool. This alarm is passed on to the operator, together with the information collected by the monitoring tool.