CHALMERS

A method for predicting hardware resource usage for Electronic Control Units

Master of Science Thesis in the Embedded Electronic System Design

EINAR NILSSON
MOHAMMED ELGHOZ

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2015

A method for predicting hardware resource usage for Electronic Control Units

Einar Nilsson
Mohammed Elghoz

Examiner: Per Larsson-Edefors

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract

The telematic gateway unit is used in vehicles for communication with a back-office. The unit is an Electronic Control Unit (ECU) and the communication could for example be used to send the current position of the vehicle, the status of the vehicle components, or the current up-time of the vehicle to the back-office. The hardware resources on the telematics gateway unit are limited and with increasing functionality of the unit increasing amounts of hardware resources are needed.

This thesis work was carried out on the onboard telematics department of Volvo Group Trucks and the main focus was to find a method to predict the required hardware resources needed when adding or changing functions on the telematics gateway unit.

After studying around 60 different articles, 9 interesting approaches were found. Regression analysis was found to be the most suited for predicting the hardware resources. The method that is proposed in the report firstly, gather data regarding the resources used on the telematic gateway unit. Secondly, multiple regression is applied to the gathered data to make the predictions of the hardware resource chosen. Limited tests were carried out on the predicted values, the tests indicated that predictions on the hardware resources were possible. The predictions made with multiple regression can be used to roughly estimate the amount of resources a new function will require. However, implementing the prediction method requires measuring more resource data than is currently done by the onboard telematics department.

Acknowledgements

This master thesis has been carried out at Volvo Group Trucks Technology. We would first of all like to thank Karin Denti and Henrik Wärmlind for giving us the opportunity to carry out this project at the Onboard Telematics Department.

We would like to express our deepest gratitude to our supervisor, Henrik Wärmlind at Volvo and our advisor, Lars Svensson at Chalmers, for the time and help they provided during the project.

Thanks also to Emil Jansson for his input during our discussions in the early stages of the project and to Liselott Kronquist for the introduction to Robot and for providing support with the test cases whenever needed.

Contents

Glossary

ANN - Artificial Neural Network

ANOVA - Analysis Of Variance

AR - Auto Regressive

CAN - Controller Area Network

CPE - Computer Performance Evaluation

DRAM - Dynamic Random Access Memory

ECU - Electronic Control Unit

EEPROM - Electrically Erasable Programmable Read-Only Memory

GNSS - Global Navigation Satellite System

GPS - Global Positioning System

GSM - Global System for Mobile Communications

HIRES - HIerarchical RESource management

LLC - Logical Link Control

MAC - Medium Access Control

MOSFET - Metal–Oxide–Semiconductor Field-Effect Transistor

OBT - On Board Telematics

OSI - Open Systems Interconnection

PCA - Principal Component Analysis

PPP - Point-to-Point Protocol

RAM - Random Access Memory

RMSE - Root-Mean-Square Error

ROM - Read Only Memory

RPS	Resource Prediction System
SRAM	Static Random Access Memory
TGW	Telematics GateWay
VM	Virtual Machine
WLAN	Wireless Local Area Network

1 Introduction

In this chapter the background to the project is presented and the problem to be solved described.

1.1 Background

In-car electronics is a fast-growing market in the automotive industry and covers everything from fuel regulation to diagnostics of problems. Cars today have between 30 and 80 separate Electronic Control Units (ECU) [14] [16]. Some of the ECUs affect safety with functions such as adaptive cruise control and lane assist systems. Other ECUs give drivers the ability to control different devices external and internal from inside the car, e.g. sound and navigation systems.

Telematics, where electronic systems and machinery communicate with each other, is a particularly important category of in-car electronics [19]. The ECUs provide specific information which for example the fleet manager can access at anytime. Telematics can be advanced, with multiple functionalities and perform valuable services. For instance, a vehicle can be tracked and/or monitored from a distance and send specific information to the remote recipient. The sent information can contain localization information, or information received through monitoring the vehicle. The received data may be used to reduce overall fuel consumption or environmental load. Another area where telematics contribute is safety [14]. By monitoring driving behavior, the fleet manager can easily for example, see when and how often a driver breaks the speed limit; in case of an accident, the telematics system can automatically send necessary information about the incident and where it occurred [8].

Fleet managers today use telematics to track where their assets are, how their condition is, and what they are doing [52]. The Onboard Telematics department at Volvo GTT provides such a service for fleet management, which uses connectivity in order to handle vehicle services such as driver assistance, fuel reduction etc., along with up-time services such as remote diagnostics and remote software downloads [10]. Volvo GTT is a part of the Volvo Group Trucks and is responsible for all the product development within the organization [17]. However, for the telematics ECU, the functional growth is expected to increase more than other ECUs in a vehicle [10]. The reason is that information is constantly being added, through usage and updates to the product. The growth puts the system resource utilization in focus, since it must be monitored so that the functional growth does not overload the ECU [10]. Through the implementation of a prediction method for the system resource utilization, it becomes possible to forecast the resource requirements. The implementation enhances decision making regarding the product's future and helps with setting appropriate requirements.

1.2 Problem description

During the ECU's lifetime, more memory is constantly required since new functions are being added and old functions are being updated. However, memory is not the only resource that is limited for the ECU; the CPU is another resource that is experiencing increased load with growing functions. Knowing when the ECU will lack memory or some other resource is required in order to adapt quickly when the hardware needs to be replaced. Something that needs to be addressed is knowing if a new function will be able to be implemented and run in a correct way.

The way memory is allocated makes a difference in order to use the available memory efficiently. Thus, the allocation needs to be reviewed and evaluated.

1.3 Purpose/Aim

The purpose of this thesis work is to provide a recommendation for a prediction method for the ECU system that can predict how much system resources is necessary and/or when the current system resource will become full. Optional goals are to investigate the use of static or dynamic memory allocation methods, and to compare them to the allocation method currently used in the ECU.

1.4 Goals/Expected result

- An analysis of resource usage predictability methods. Included in the analysis is performing a literature study of different methods.
- Recommendation for OBT (On Board Telematics) regarding suitable predictability methods.

- Simple proof of concept, which covers verifying that the identified method works.

- Review of different methods and theories for dynamic resource allocation. (opt.)

- Gap-analysis for OBT regarding dynamic vs resource allocation methods. (opt.)

1.5 Limitations

This section describes the limitation of this thesis work.

- One of the desired goals from Volvo is to provide a state-of-the-art analysis on resource predictability methods. However, it is doubtful that there will be enough time to perform such a detailed and extensive analysis on multiple prediction methods and thus, it has been changed to literature study and thorough review.

- Due to constant changes, such as new drivers and updates for the product, more memory is constantly required. Furthermore, the CPU is constantly in use. Therefore, the main focus will be on memory and CPU usage, although it is possible to use and find other measurable data.

- Since the main focus will be hardware resource predictability, the existing ECU system will be analyzed, however no alternative hardware will be analyzed regarding the current system.

- The optional goals are included in the time-plan since we aim to complete, but will be removed if time becomes short.

- In case access to tools to perform a proof-of-concept becomes limited, then that part will be neglected and the analysis part will be carried out more extensively.

- Since the ECUs resources are limited, the amount of programs continuously run needs to be kept low. For this reason the resource predictability method should not be implemented as a function on the ECU.

1.6 Thesis Structure

The content of the thesis is structured as follows: Chapter 2 is where the work process and resources are described. In Chapter 3 the Telematics gateway (TGW) and its resources are described. Regression analysis and neural networks, which are the methods considered for prediction, are also described in Chapter 3. The literature study performed and methods reviewed are described in Chapter 4. Chapter 5 explains how the methods are analyzed and how the recommended method is carried out. The test results are provided and explained in Chapter 6. Chapter 7 is where the work process, results, and the prediction are discussed. Chapter 8 concludes the thesis and discusses future work.

2 Methods and materials

This chapter describes the approach used to carry out the project, the different materials used during the project, and detailed descriptions of the tests done.

2.1 Procedure

Planning and scheduling different parts of the project was done initially in order to provide an ordered structure for the entire project. Thus, the project was divided into different phases:

Planning
During this phase, the initial plan of the project will be made. The scope, limitations, risks and initial time plan will be determined. This phase is considered to be finished when the planning report is approved.

Information gathering
In the second phase, information regarding the Telematics GateWay (TGW) will be gathered and analyzed. Furthermore, information will also be gathered regarding prediction methods. A first analysis of the material gathered will be done in order to sort the information.

Analysis
After the information gathering is completed, a thorough analysis about the product and prediction methods will be carried out. The critical areas of the TGW must be singled out and a model for prediction that is deemed implementable will be selected.

Recommendation and implementation analysis
A recommendation of applicable prediction methods will be made, together with and an analysis on how the method would perform and/or affect the system.

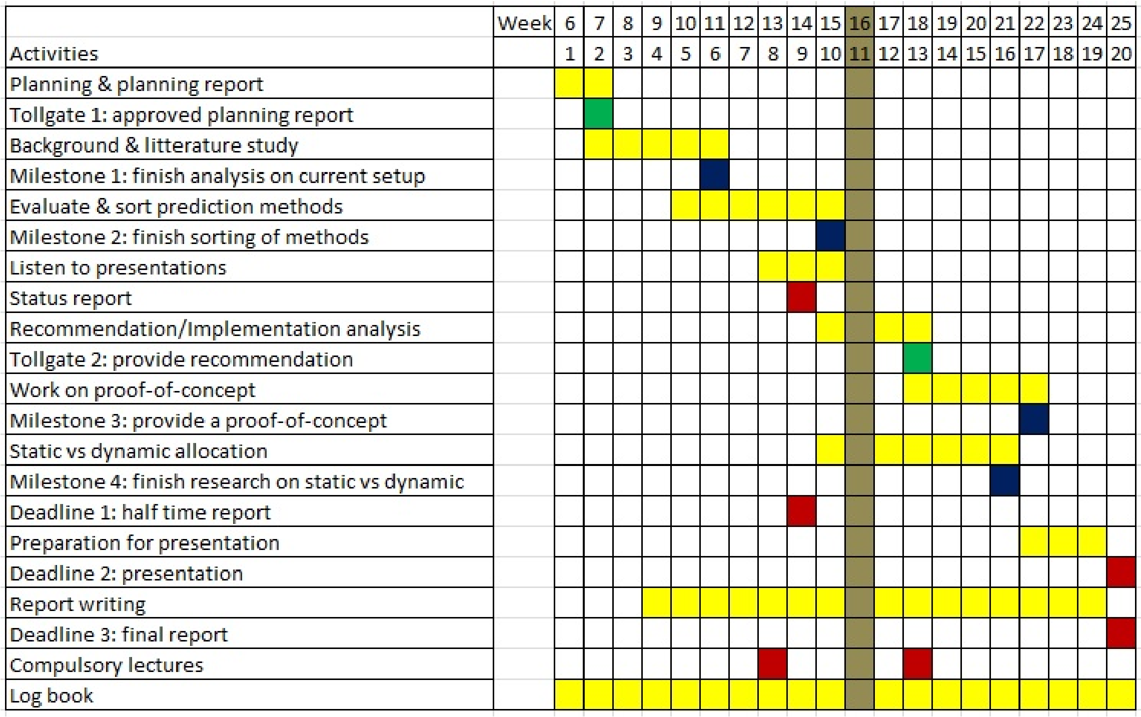
Verification
A prototype of a selected prediction method will be made and testing will occur in order to verify the ability to predict and the prediction accuracy.

Finishing the project

A proposal of an appropriate prediction model will be made. This phase is concluded with the finishing of the final academic report and the presentation.

2.2 Timeplan

The initial timeplan for the project can be seen in Figure 1 below.



	Week	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Activities		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Planning & planning report		🟨	🟨									▨									
Tollgate 1: approved planning report			🟩									▨									
Background & litterature study			🟨	🟨	🟨	🟨	🟨					▨									
Milestone 1: finish analysis on current setup							🟦					▨									
Evaluate & sort prediction methods						🟨	🟨	🟨	🟨	🟨	🟨	▨									
Milestone 2: finish sorting of methods											🟦	▨									
Listen to presentations									🟨	🟨	🟨	▨									
Status report										🟥		▨									
Recommendation/Implementation analysis											🟨	▨	🟨	🟨							
Tollgate 2: provide recommendation												▨		🟩							
Work on proof-of-concept												▨		🟨	🟨	🟨	🟨	🟨			
Milestone 3: provide a proof-of-concept												▨						🟦			
Static vs dynamic allocation											🟨	▨	🟨	🟨	🟨	🟨	🟨				
Milestone 4: finish research on static vs dynamic												▨					🟦				
Deadline 1: half time report										🟥		▨									
Preparation for presentation												▨						🟨	🟨	🟨	
Deadline 2: presentation												▨									🟥
Report writing					🟨	🟨	🟨	🟨	🟨	🟨	🟨	▨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
Deadline 3: final report												▨									🟥
Compulsory lectures									🟥			▨		🟥							
Log book		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	▨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

Figure 1: The initial timeplan for the project.

2.3 Resources

The following materials will be used during this project:

- TGW (Hardware) - used for testing.
- TeraTerm (Software) - used to communicate with a TGW.

- Office (Software) - used for writing report (Word) and as a tool capable of performing regression analysis (Excel with Analysis Toolpak, an add-in tool).
- R (Software) - Quick R is a statistical and graphical programming language.
- Rstudio (Software) - an open source user interface for R.
- Robot framework (Software) - used to create test cases, that are used for collecting data for predictions. Robot requires the following software applications to function:
  - Python 2.7 (Software) with pySerial (Serial port logger, python language extension) and Win32Com (Communication client, python language extension).
  - CTA (Communication Test Application)
  - CANoe (Used to simulate a truck environment)
  - ClearCase (Used to store the test cases and other code)

2.4 Statistical procedures used

This section describes tests and comparisons that will be used in Chapter 6.

2.4.1 Test cases

The TGW is tested by running different test cases, which simulates different functions on a TGW and records the simulation results. The test cases are run using the Robot framework, CTA, CANoe and a TGW. The Robot framework used for testing is based on the Python language, but uses keywords. The keywords executes python code when written. Test cases are run to test the functionality of the TGW and some of the test cases are used for overnight testing done regularly.

Two test cases for testing and for measuring the resource levels for the TGW was to be created, one for heavy load and one for idle load. These test cases were to be used when measuring the resources.

The heavy load test case was created to test the TGW when running many function that requires a lot of resources. While running the heavy load test case, the resources were measured once just after all the functions were started and once 10 minutes afterwards. The idle load test case simply started the system and measured the resource usage after 10 minutes of running the system.

3 Technical background

This chapter describes the theory behind several of the components studied to use for resource predictions and general descriptions of the area in which the predictions are to be applied to. Methods on how to predict resources are also presented in this chapter.

3.1 Telematics

Telematics are usually used in the automotive industry as a way to transmit real-time data back to an organization [15]. The telematics can for example be used to collect information about how a vehicle is used or its maintenance requirements and provide real-time information about the vehicle. GNSS (Global Navigation Satellite System), which for example could include GPS (Global Positioning System), and GSM (Global System for Mobile Communications) are tools that are used by telematics to keep track of the location of a vehicle. Fleet managers today can, with telematics connected, easily see where their vehicles currently are, if they are on or off the road and what state they are currently in [52]. This information may also be available on mobile devices for the fleet manager, allowing surveillance of the vehicles to be done even more often.

The history of telematics can be seen to start in 1984 when the European Parliament passed a resolution to improve road safety [36]. The resolution resulted in a research program that focused on using telematics to improve road safety and to reduce the environmental impact of vehicles. In 1993, a European treaty was signed, which declared that Europe's global competitiveness could be increased through stronger logistical networks and transportation. At this time, information and technology was seen as important components for success and leaders were hoping to achieve increasing global competitiveness by using telematics. In the 1990s, GPS systems became more available for the public and was used for the telematics market [19].

3.2 Electronic Control Unit

ECUs are used in the automotive industry to control electrical subsystems in vehicles [18]. Almost every subsystem in a vehicle has an ECU and modern cars can contain around 80 ECUs. ECUs are used for infotainment, climate control, engine control, transmission, body (e.g door locks, windows) and more [9]. The advantages of ECUs are reduced exhaust emission, better engine operation with longer lifetime, and efficient fuel usage. More, and better, safety functions can be implemented, as well as comfort functions and diagnostics tools. In short, ECUs provide improved drive-ability. For example, real-time regulation of fuel injection and ignition timing can guarantee maximum engine performance while keeping the fuel consumption to a minimum. ECUs also improve the after-treatment systems for the exhaust gas.

An ECU consists of a power supply module, a system clock, a microprocessor/microcontroller, an I/O interface and a memory unit [9]. The microprocessor/microcontroller is used to execute code and process information through different control algorithms in order to generate output signals. I/O interface is used to register operating conditions and desired values through sensors, and uses actuators to convert electrical output from the ECU to mechanical parameters. The memory unit consists of boot memory, program memory, calibration memory and data memory.

3.3 Telematics GateWay

The Telematics Gateway (TGW) is an ECU used by Volvo in vehicles [46]. The TGW is used for storing data and communicating with off-board systems. The main functions of the TGW are: to send vehicle data to an office portal, to log vehicle data (including fuel consumption, position, and driver data), and to be a gateway for remote services (e.g. software download or remote diagnostics).

The TGW uses a FRAM memory, a SDRAM memory, and a NOR Flash memory [49]. The protocols used by the TGW are generally described in chapter 3.7 and are the following: CAN, WLAN, J1708/J1587, Ethernet, and PPP. The CPU is an ARM 9 and uses a modified form of the round robin scheduling algorithm to schedule the majority of its tasks.

3.4 Memory

As described in Chapter 1, the memory requirements is one of the major target for our prediction efforts. Several memory technologies are of interest in this work; this section gives a technical overview.

3.4.1 Flash

Flash memory is a non-volatile storage technology and originates from EEPROM (Electrically Erasable Programmable Read-Only Memory)[35]. They both use electrical charges to erase data [43]. However, flash memory is generally faster than the EEPROM, since it erases a block or an entire chip instead of one byte at a time. The fact that it is non-volatile means that the data is not lost if the flash memory loses power that is one of its advantages. On the other hand, some of the disadvantages are that it is slower than other forms of memory, such as RAM. The flash memory also has a limited number of write/erase cycles, and in order to write new data previous data must first be erased. Another factor that is both an advantage and a disadvantage is that it erases and writes in blocks. This allows it to operate faster than an EEPROM for example, but it might not be practical in every situation.

There are two types of flash memory, NAND and NOR [35]. The NAND flash memories are accessed like block devices (e.g. hard disks). In order to read from NAND flash memories, the data must be paged to a memory-mapped Random Access Memory (RAM). Thus, a memory management unit is required. NOR flash memories, allow for reading of individual flash memory cells, which makes its behavior similar to a ROM (Read Only Memory). Commands for write and erase functions are written to the first page of the mapped memory.

3.4.2 RAM

Random Access Memory or RAM is a memory that allows any cell to be accessed or written to without accessing other cells in the same area of the memory [5]. There are two basic types of RAM, DRAM (Dynamic Random Access Memory) and SRAM (Static Random Access Memory). Both of these types are volatile and lose their data once power is lost.

The DRAM stores each bit of data in a separate capacitance [20]. The electrical component that stores the data for a DRAM is one capacitor and one transistor per bit. The data on the electrical component needs to be refreshed regularly or the stored data will be lost. The refreshing of the data takes time and while this is done no access to the memory can be done.

The SRAM will store the data for as long as it has power without refreshing it [21]. The electrical components used to store one bit in the classical SRAM are six MOSFETs (Metal–Oxide–Semiconductor Field-Effect Transistor). The SRAM is faster than the DRAM, however it also consumes more energy.

3.5 Central Processing Unit

The CPU is where most calculations are done [11]. A CPU usually consists of an arithmetic-logic unit, which is where logical and arithmetic operations are performed; and a control unit that extracts instructions from memory which is then decoded and executed. The CPU is limited in the amount of instruction it can handle per second. Thus, computations is a limited resource and the requirements for it increases over time together with the memory requirements.

3.6 Scheduling

The scheduling of the processor affects the amount of resources that is required. Depending on the scheduling algorithm the tasks scheduled will have to wait different amounts of time to be executed.

Scheduling is used to assign a process from the ready queue to the CPU, when the CPU is idle [38]. There are several different scheduling algorithms, preemptive algorithms where a process can be emptied out of the CPU to make room for another process and non-preemptive algorithms where only the process using the CPU can decide when it is done executing. Scheduling criteria that are affected by the scheduling algorithm is the CPU utilization, Throughput, Turnaround time, Waiting time, and Response time.

3.6.1 Round Robin scheduling

The Round Robin scheduling algorithm works by cycling through a list of queued jobs [34]. Each one of the jobs has an assigned time slot in which it will be executed before the processor changes to the next job in the time-slot lists. The algorithm makes sure that each job receives a time slot regardless of the time required to finish executing the job, or any other characteristics of the job. Once the time-slot is used up, the job will have to wait for the next time its time slot is up for execution. The list containing the jobs is a FIFO queue and new jobs are added at the end of the queue [24]. By increasing the time slot for a task or by adding the task more than once in the queue the task will receive a higher priority.

3.7 I/O devices and protocols

In this section, we present the I/O devices and protocols considered during the project.

3.7.1 OSI-model

The OSI (Open Systems Interconnection) model is a framework which defines how network protocols are to be implemented in seven layers and the interactions between the layers[4].

Table 1 describes the OSI layer model and shortly explain the function of the different layers. The OSI layers are later in the chapter used to describe what layers the devices and protocols applies to.

Table 1: Overview of the OSI layers [31]

Layer:	Function:
7 Application	Provides user interface
6 Presentation	Presents data, processing, encryption of data
5 Session	Keeps different applications data separate
4 Transport	Provides reliable or unreliable delivery, preforms error correction before transmit
3 Network	Provides logical addressing for path determination
2 Data link	Error detection, combines packet, combines frames, provides access to media using MAC address
1 Physical	Moves bits in devices, specifies voltage, wire speed, and pin out of cable equipment

3.7.2 CAN

CAN is short for Controller Area Network and is a serial communication protocol that for example is used in automotive electronics [1]. The CAN communication can reach bit-rates up to 1 Mbit/s. The CAN is divided into different layers in the OSI model: the Data Link Control, which has two sub-layers, and the Physical layer. The sub-layers are called Logical Link Control (LLC) and Medium Access Control (MAC). The LLC sub-layer provides services for transferring data. It decides which messages are sent to the LLC that are to be accepted and also makes it possible to manage recovery and receive overload notifications. The MAC sub-layer manages the transfer protocol and handles the buses. The Physical layer sends bits between different nodes in the network.

3.7.3 WLAN

A Wireless Local Area Network (WLAN) allows wireless communication using radio signals [30]. The WLAN communication replaces the cables when connecting to a network by creating a wireless access point on the wired network. Mobile devices can then connect to the network using the access point. WLAN provides functionality within the Physical layer and in the the Data Link control layer in the OSI model [7].

3.7.4 J1708

J1708 is a protocol for serial communication developed for heavy vehicles to minimize the hardware costs [26]. J1708 only applies to the Physical layer and the Data Link layer in the OSI model. The transmission rate for the protocol is 9600 bps and the transmissions can be up to 21 bytes long, with error detection and collision handling.

3.7.5 J1587

The J1587 protocol is usually used with the J1708 protocol and is used in heavy vehicles [25]. The J1587 protocol is used to exchange data from different parts of the vehicle to where the data is used. J1587 is in the Application layer and describes the message format and allows the creating of connections to be able to send more than the 21 bytes the J1708 is limited to.

3.7.6 Ethernet

Ethernet is a local-area-network technology which is used to connect devices in close proximity [33]. Ethernet devices are attached to a common medium which provides a path for the signals to travel. A single shared medium is called an Ethernet segment; devices attached to a segment are called nodes. The communication among the nodes is carried out through differently sized chunks of information called frames. The Ethernet protocol has a specific set of rules for the construction of frames. For example, there are defined minimum and maximum lengths for frames, and some pieces of information must exist in the frame, e.g. destination and source addresses. Ethernet uses isolated segments to prevent collision, and older versions of the twisted-pair Ethernet used carrier-sense multiple access with collision detection (CSMA/CD) which was used to describe how the communication between nodes is regulated [47]. In short, whenever a station wants to transmit, it listens to the medium. If no other station is transmitting then it starts transmitting [33]. Collision occurs when more than one station is transmitting at the same time. When this happens, the station terminates the transmission and waits a random amount of time before trying again. Ethernet operates within two lower layers of the OSI model, the physical layer and the data link layer [6].

3.7.7 Point to point protocol

PPP is a point-to-point network protocol and is used to send data between two directly connected points (computers)[22]. The protocol links two computers together and allows data to be transferred directly between them, the protocol is often used when the connection has to be fast and transfer a lot of data. PPP is a part of the Data Link control protocol in the OSI and is designed to work for different Network Layer protocols [39].

3.8 Regression analysis

This section describes how to perform a linear regression analysis, both with single and multiple variables, as well as related calculations that assists in prediction accuracy.

3.8.1 Pearson's correlation coefficient

Correlation is used to determine how well different sets of data are related [3]. Pearsons Correlation shows the linear relationship between two sets of data. The equation is as follows:

$$r = \frac{n(\sum XY) - \frac{\sum X * \sum Y}{n}}{\sqrt[2]{(\sum X^2 - \frac{(\sum X)^2}{n})(\sum Y^2 - \frac{(\sum Y)^2}{n})}} \tag{1}$$

where X and Y are the variables for the data sets and n is the number of data points in each set. The result of the equation will always amount to a value between +1 and -1. The value 0 signals that there is no correlation. If a value is below zero it signals a negative correlation and a value above zero signals a positive correlation. A positive correlation means that the data sets increases and decreases together while a negative correlation means that a value increases while the other decreases. A drawback with the Pearson correlation method is that it cannot tell the difference between independent and dependent variables. In short, this method only tells if there exists a relationship, the rest is up for interpretation.

3.8.2 Coefficient of determination

This coefficient is interpreted as the level of variance in the dependent variable that is predictable from the independent variable [41]. If the value is zero, it means that the dependent variable is not predictable through the independent variable. If the value is one on, then the dependent variable can be predicted from the independent variable without error. The equation is defined as:

$$r = \frac{\sum (xy)}{\sqrt{(\sum x^2) * (\sum y^2)}} \tag{2}$$

where x is equal to:

$$x = x_i - \tilde{x} \tag{3}$$

x_i is the value x for observation i and $\tilde{x}$ is the mean value of x. The variable y in equation 2 is equal to:

$$y = y_i - \tilde{y} \tag{4}$$

where y_i is the y value for observation i and $\tilde{y}$ is the mean y value.

3.8.3 Linear regression

Simple linear regression is used to predict values on one variable from the values of another variable [27]. Y is the variable that will be predicted and is called the criterion variable. The variable that the predictions are based on is referred to as X and is called the predictor variable. The equation for simple linear regression is as follows.

$$Y = bX + A \tag{5}$$

where b is the slope of the line and A is the intercept of Y.

3.8.4 Standard error of estimation

The standard error of estimation measures the accuracy of the predictions and is defined as follows [28]:

$$\sigma = \sqrt{\frac{\sum (Y - Y")^2}{N}} \tag{6}$$

where σ is the standard error of estimate, Y is the actual score, $Y"$ is the predicted score, and N is the number of value pairs.

3.8.5 Multiple regression

Multiple regression is used for predicting an unknown value for a variable from multiple variables with known values [13]. The variable with the unknown values is referred to as the dependent variable while the others are referred to as the independent variables. The general multiple regression of Y (the dependent variable) on X_1,X_2...,X_k is given by:

$$Y = b_0 + b_1X_1 + b_2X_2 + \cdots + b_kX_k \tag{7}$$

where b_0 is the intercept and b_1,b_2,...,b_k are called regression coefficients. The b variables are called "beta weights" and these values represent the change in the Y-variable correlated with a change of the standard deviation on a predictor. Different parameters may have different impact on the result. This also means that one parameter may have a small impact on the end result alone. However, together with another parameter, which it is highly correlated with, it may have significant impact.

3.8.6 Multiple determination coefficient

In multiple regression, the proportion of variance is described as R^2 [50]. It is the quotient of the variances of the fitted and observed values of the dependent variable. The coefficient of determination is defined as follows:

$$R^2 = \frac{\sum(\hat{y}_i - \tilde{y})^2)}{\sum(y_i - \tilde{y})^2} \tag{8}$$

where y is denoted as the observed values of the dependent variable, $\tilde{y}$ as its mean and $\hat{y}_i$ as the fitted value. The difference between R^2 and the equation described in chapter 3.8.2 is that R^2 gives a value that includes the correlation for all the used variables on the variable that is being predicted.

3.9 Neural network

Neural network was considered as an alternative to regression analysis. A neural network does not follow instructions in a linear fashion, but processes information collectively [37]. This is done in parallel throughout a network of nodes. The nodes are in this case called neurons. A neural networks ability to learn is one it its key elements. Since it is a complex adaptive system, it has the ability to change and reconstruct its internal structure based on the information being handled through it. The individual element of the system, the neuron, takes input/inputs and processes it in order to generate an output. The change of the systems structure is possible through the adjusting of weights. Each connection between two neurons has a weight, which is a number that controls the signal between the neurons. If the network generates a positive output, then nothing happens. On the other hand, if a negative output is generated, then the system adapts by altering the weights in order to improve the following results.

The system can learn in different ways. An example is supervised learning, in which the correct answers are known. The system can then compare its generated results with the correct ones in order to make adjustments. There is also unsupervised learning where a set of elements are divided into groups according to an unknown pattern, i.e. clustering. A third method is reinforcement learning, which is based on observation. In this case, the network makes a decision with an outcome and observes the environment. If the observation is negative, then the weights are adjusted in order to make a different decision next time.

3.9.1 Perceptron

The simplest form of a neural network is a perceptron, which is a computational model for a single neuron. The perceptron consists of, as described in section 3.9, of one or multiple inputs, a processor and a single output as seen in figure 2 [37]. It follows a model called "feed-forward" which means that inputs are sent in, are processed, and result in a single output. An input that is sent into a neuron must first be weighted, which means it has to be multiplied with a value signaling its weight. The weighted inputs are then summed, and sending that sum through an activation function generates the output. In case of a binary output, the function decides whether the perceptron should trigger or not. The activation function can be designed in multiple ways and can be simple, e.g. comparing the sum to a predetermined value, or they can be more complex with advanced algorithms.

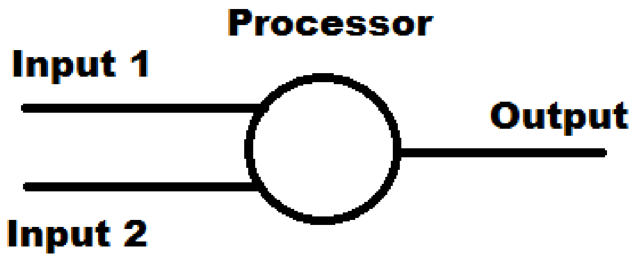


Figure 2: An example of a perceptron.

3.9.2 The network

The network consists of multiple and connected neurons, i.e. a multi-layered perceptron [37]. In a setup like that, the neurons serve different purposes. Some neurons serve as input neurons and some are part of a "hidden" layer since they are not directly connected to an input or output, as seen in figure 3. Finally, there are the output neurons from which results are read. In a simple neuron it is easy to adjust the weight depending on the output. A complete network is trickier since there are so many connections in different layers of the network. In order to optimize the weights of a multi-layered network, back-propagation is used. Here, output is generated in the same way as for a single perceptron. The difference is that the weighted inputs travel through additional layers of neurons before reaching the output. Training the network involves taking the error as well and feeding it backward through the network. The final error adjusts the weights of all the connections.

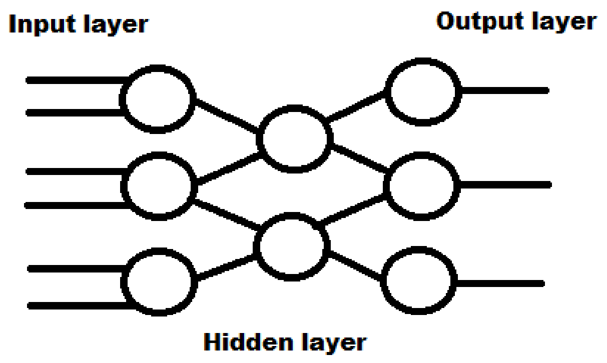


Figure 3: A simple network with an input layer, a hidden layer and an output layer

4 Prediction methods

This section briefly describes the most interesting among the almost 60 articles and books studied for achieving the goals concerning predicting resource usage described in Chapter 1.4.

4.1 HIRES

HIRES is short for Hierarchical Resource management and is a method presented by Parmer et al for predicting resource management in a hierarchical system [32]. HIRES focuses on predictable hierarchical resource management. The resource management is achieved by distributing the available resources to different processes that in turn divides the resources required for different functions to child processes. The child processes could in turn create their own child processes to distribute the assigned resources even further; this was proven by Parmer et al to increase the resource management flexibility. HIRES ensures that the resource usage and the control of the resources remain constant regardless of the hierarchical depth, making the hierarchical resource management more predictable.

4.2 Two layer queuing

Predicting performance in the presence of bottleneck circumstances for both hardware and software resources may be done by introducing a two layer queuing system as showed by Duttagupta et al. [12]. The two layer queuing system is used to achieve predictability by handling requests with critical sections and pooling of resources in multi-classes. The accuracy of the method is validated by running real experiments and comparing to the predicted values. The experiments included running sample java programs containing critical sections, with varying number of users, on three different types of servers. Three different scenarios were considered: one where the critical sections and the non-critical section are almost the same size, one where the critical section is twice the size of the non-critical section and one where the non-critical section is twice the size of the critical section. During the test the average throughput, the response time of the program and the CPU utilization where measured. Duttagupta et al. compared the measured values to the predicted values and the results proved that the method was able to predict throughput and resource utilization with accuracy close to 90%.

4.3 Forecasting resource requirements

Sinuany-Stern et al. present a model to evaluate the utilization level of computer hardware resources, to be able to find the current needs and predict future needs for a computer center [40]. The manager of the computer center wants to be able to predict which resources need to be improved and which resources can remain unchanged. The model is based on analyzing the performance data of the hardware resources. It can predict the the amount of resources needed when increasing other resources and use these predictions to identify bottlenecks.

Depending on the job that is run on a computer, the resource requirements change. Some reasons for this are: that different jobs use different operations and different amount of the resources; that the number of operations that needs to be performed depends on the input data; and that if multi-programming is used, a job may require different amounts of the resources depending on the other jobs executed at the same time. Most of the time the consumption of the different resources are statistically related to each other. By using this relationship, given one or more consumption level for the resources, it is possible to predict the other unknown consumption levels for the other resources using *regression analysis*. The reliability of the prediction improves with a larger number of data samples for the different resources.

The model presented by Sinuany-Stern et al. uses regression analysis to predict the average usage level for a resource. The following equation is used for predicting one future resource:

$$\hat{x}_{j/i} = A_{ij} + B_{ij}x_i \tag{9}$$

Where $\hat{x}_{j/i}$ is the level of the predicted resource j given resource i and A_{ij} and $B_{ij}x_i$ are the regression coefficients. The predicted level of the resource j is called E_j and is calculated using the following equation:

$$E_j = U_j + (A_{ij} + B_{ij}x_i)K_p = U_j + \hat{x}_{j/i}K_p \tag{10}$$

Where U_j is the current level of resource j, and K_p is the number of additional jobs. Bottlenecks are then identified by calculating the utilization (g) and comparing it to the maximum level of utilization. The utilization is calculated using the C_j, which is the maximal consumption level of resource j, and using the following equation:

$$g = (U_i + \hat{x}_{j/i}K_j)/C_j \qquad \forall j = 1, ...J \tag{11}$$

The number of jobs that is added is then calculated using the following equation:

$$K_j = (C_j g - U_j)/\hat{x}_{j/i} \tag{12}$$

The critical number of jobs will be the minimum value for K_j, and any new K_j values can be compared to the critical value to decide if new resources are needed for the system.

When more than one resource is to be predicted the following multiple regression equations are used:

$$\hat{x}_{j/J_1} = (A_{J_1 j} + \sum i \in J_1 b_{ij} f_i(x_i))^{-f_j} \qquad \forall j \in J_2 \tag{13}$$

$$\hat{x}_{j/J_1} = x_j \qquad \forall j \in J_1 \tag{14}$$

Where J_1 includes a set of resources where the future average amount per job is known, and J_2 includes the set of resources where the future average is unknown and is predicted in the equation. $A_{J_1 j}$ is a regression constant from the regression of x_j, b_{ij} is a regression coefficient, and $-f_j$ is used to inverse transform the result to make it usable. The bottleneck is then identified in the same way as for the single regression when the average resource level for a job has been predicted.

The resources analyzed, by the computer center, that are used as variables in the regression analysis are CPU, I/O and Memory. For the CPU, the amount of time that each job uses the processor was measured, for the I/O the number of disk inputs and outputs for each job was measured, and for the memory the amount of memory used for each job was measured. When collecting the data, the sampling was done systematically every hour during the sampling period where all tasks performed during one minute were listed. With the collected samples, a wide range of scenarios were run and the predictions pointed towards that memory was most often the cause of bottlenecks. With these results, the manager of the computer center knew that memory was the resource that was in greatest need of being improved. Finally Sinuany-Stern et al recommended that the coefficients and the usage level of each resource were updated periodically for most accurate predictions.

4.4 Resource management

Zhang et al have studied a cloud computing service [53]. The study provides a flexible method for maintaining an appropriate size of the resources. A system in three parts is proposed: a workload prediction module, a workload balancer, and a resource manager. The different parts are used to manage the workload balancing and for the resource management. The workload balancing and resource management is done using a controller node built by Zhang et al. The controller node is connected to several cloud compute nodes.

A node consists of two modules: the workload balancing module and the resource managing module.The workload balancing module makes a workload prediction that is sent to the resource managers and used during the next scheduling period. The predicted workload is then used by the resource manager to adjust the virtual machine (VM) resources. The resource manager sends VM information to the workload balancer to receive a reply on how to make the adjustments to balance the workload in the best way. The workload balancer uses the information about the VMs and decides the most suitable VM for a request based on the CPU utilization, I/O utilization, and the internet condition of the VMs.

Predicting the workload is done using a statistical method, which uses historical data. Data about usage patterns was collected during a 24-hour periods by Zhang et al. The data was then divided into several sections. Based on the reasoning that similar businesses also will have similar workloads, historical data from a research data-set was used. The data-set contained several records and by using information theory, the probability and uncertainty for the different records were calculated. The record with the highest reliability was then chosen to be used for the predictions. When a period has ended the workload records from the past period has to be processed in the workload prediction center. If the data are new, a new record has to be created in the historical information. A new record is also created if the data deviates too much from previous record of the same type. However, if the record does exist and does not deviate too much, the predictability and uncertainty is updated. Records with lower uncertainty will be chosen before records of the same type with higher uncertainty.

Zhang et al tested their prediction model by simulating 1000 users visiting a cloud cluster containing one controller node and five compute nodes. When comparing the predicted values to the measured values, Zhang et al. argue good correspondence. The method described in the study could improve the performance of cloud clusters by making the resource management more effective.

4.5 Resource usage patterns

Predicting resource usage patterns for the near-future have provided support for improvements when scheduling processes, managing disk I/O and in quality of service [51].

Yoas et al [51] have observed the use of business prediction models to predict long-term resource utilization for servers, since the business prediction focuses on human interaction patterns. This model may also be usable when predicting computer resource requirements.The business model has been used to predict different resources, traffic flow and short-term water usage for example. The predictions given enough history can be made for hours, months and yearly trends. Yoas et al evaluated predictions based on human-generated use of applications and systems. The results supported the possibility of using human-generated patterns to predict the computer resource utilization. Examples of how these kinds of patterns can be used for predictions is when predicting an increase in phone usage on Mother's Day, or electricity utilization for air conditioning during the summer.

To test long-term prediction for server resources, a server was used to collect data about the CPU utilization, available memory, network traffic, and disk activity. The data was collected in a log every ten seconds, and after 24hours, a new log was started. The test used twenty-two client computers and the activity on the server was simulated using a Scalable URL Reference Generator (SURGE). A tool called SAS [48] was used to examine the data and was then used to generate a pattern. The data from the pattern was then compared to the measured data. Linear regression was then used as an additional evaluation and received a confidence level of 80% [51]. By using patterns, and their influence over the resource utilization, Yoas et al found it to be possible to predict resource utilization for servers accurately further into the future than the related work mentioned in the study. However common human generated patterns within a system must first be identified.

4.6 Predicting through patterns

Combining VMs is an important technique used in cloud computing to increase the resource usage [42]. To estimate future resource demand for each VM, historical workload traces are used when characterizing the resource usage patterns. Two methods for predicting the resource consumption are presented by Tan et al [42]. The first method predicts the resource usage for a specific node and uses both CPU and memory usage for its performance predictions. The second method focuses on a large number of nodes and the workload running on them, which is likely caused by a smaller number of jobs. Tan et al propose an approach based on PCA (Principal component analysis) to predict the CPU and memory usage in the commercial data center. Most approaches ignore that the workloads vary, and for a specific node the workload may depend on several other nodes. Tan et al propose a solution where VMs are placed so that the variability on the server is reduced.

The data used by Tan et al was collected from IBM global services and is collected during a 3 month period from 15883 VMs hosted on 1323 physical servers. The data from the VMs included utilization ratio for the VM´s CPU and memory. The VMs had varying levels of resources and load patterns. To reduce the periodicity exhibited by the VMs an ARAR algorithm was applied. The ARAR algorithm works by applying a memory shortening method to the data and then apply AR to make its predictions from the data [44]. AR (Auto Regressive) models were then applied for the two different methods, one model with a single dimensional AR and one model for a multi-dimensional AR model [42]. To evaluate the coefficients of the models, the Yule-Walker method was applied and Root Mean Square Error (RMSE) between the predicted value and the measured values was done to decide the accuracy of the predictions.

Tan et al noted that for CPU predictions two single-dimensional AR is the better choice, while for memory predictions the multi-dimensional AR is the better choice for the most accurate predictions. Tan et al also investigated problems related to predicting the resource usage for one specific node. Firstly they wanted to identify the resource usage patterns for the whole data center and then use an AR model prediction on the components responsible for most variability. The predictions was then tested and for the CPU the prediction error was equal to 31.3%, which can be compared to using the two single-dimensional ARs on a VM, where the prediction error was equal to 32%. Tan et al claims that by identifying the resource usage pattern the number of parameters for the model is reduced without having to sacrifice much errors when predicting the nodes separately.

4.7 Computer resource consumption

This model by Niv Ahituv and Magid Igbaria aims to predict and evaluate computer resource consumption [2]. This methods uses a metric called Computer Performance Evaluation (CPE) and describes it as an essential in the process of computer selection, configuration design and controlling existing systems. The focus in this method is the prediction and evaluation of computer resources with the use of statistical methods in order to improve the performance of existing systems and determine how well a specific system is meeting or may be expected to meet future workload .

CPE is divided into three main areas: computer selection, computer design and configuration improvement (tuning and optimization). The objective is to optimize the response of an actual or projected workload. CPE attempts to determine how well a specific system is or could meet specific processing requirements under given or planned resources.

This method uses a software monitor to collect data on the performance of each hardware component. Some performance variables, such as CPU time, might, due to the nature of multi-programming and the virtual storage, be affected by the random mix of programs residing in memory at any moment. Second, programs are not deterministic in the sense that they contain IF statements. Thus, the execution of different code segments is conditioned upon specific input contents. Third, computer setups usually host a variety of different programs, and each is likely to have its own profile of hardware resource consumption.

Each run of a computer program can be printed to a list (a vector) of actual performance variables: CPU time, number of accesses to disk drives and number of input and output records. Thus, multiple runs of the same program will likely differ in the performance vectors. It could be that some programs maintain a common profile of resource consumption, for example, all the programming assignments of first year students. The profile can then be checked by sampling a large number of performance vectors and running a large linear regression on the vector elements. E.g. if the number of input transactions are known, a prediction about the consumption of CPU time, the number of channel activities and the number of printed lines can be done. The results will then indicate whether the consumption of a resource is correlated with another resource. If the results indicate high correlation between the resources then the installation is a homogeneous one, otherwise it is called heterogeneous.

It is possible to identify a number of homogeneous groups in a heterogeneous case. By constructing a set of simultaneous equations, it is possible to identify the relationships between the consumption of various resources and use these equations to model and analyze the computer resource consumption. Thus, one can build coefficient matrices that indicate the relationships among the consumption of various computer resources within and between various groups of users. The model presented elaborates on these relationships and how they can be used for managerial purposes. Linear regression is used to predict the future utilization of computer resources. However, it is necessary to know two input factors: the hardware configuration of a specified system and the required capacity of existing applications.

Furthermore, homogeneous groups must be defined. Such groups are characterized by jobs/tasks that are relatively similar in the pattern of their resource consumption and are usually functionally related. The processing capacity requirements should also be designated. The typical information necessary is the workload to be processed and turnaround times to be met by either month, week or day.

The future utilization (F_U) of the computer resources consists of the current utilization (C_U) of the given workload and the additional utilization (A_U) derived from new requirements:

$$F_U = C_U + A_U \tag{15}$$

$$A_U = f(\underline{X}, L) \tag{16}$$

where $\underline{X}$ is a vector of predictors and L is the group to which $\underline{X}$ belongs. The purpose is to predict the additional utilization given both the set of predictors $\underline{X}$ and the group of jobs L to which they belong (L is assumed to be relatively homogeneous). For example, one may choose to forecast CPU time based on the number of disk input/output operations and the number of jobs in a specific group such as student jobs in a business school. Answers are sought to a variety of questions relating to future utilization given at least one independent variable and a group name. Since interest lies on quantitative evaluation, a statistical model is viewed as a set of mathematical formulas and quantifiable workload parameters. The model:

J - the number of hardware resources treated by the model (e.g. CPU, memory, disks, etc.).

M - the number of homogeneous groups. Each system consists of a set of groups that are relatively similar in the pattern of their resource consumption.

C - Vector of resource capacities, e.g denote the daily capacity of CPU time

U - Vector of resource utilization, e.g. denote the number of disk I/O operations during a predetermined time.

B - Matrix of the slope coefficients of the independent variable. B defines the relationship between two or more computer resources. B includes M sub-matrices for each group of jobs. B_{ij}^{m} denotes the slop coefficient of X_i to predict the dependent variable X_j in group m.

A - Matrix of the intercept coefficients. Includes M sub-matrices for each group of jobs. A_{ij}^{m} denotes the intercept coefficient of the independent variable X_i to predict X_j in group m.

P - A column vector of $J \times M$ elements that contains the expected additional amount of a given hardware resource. If the future consumption of only one independent variable is given, then P will be a vector of zeroes, except in the corresponding element that defines the additional amount.

I - A column vector of $J \times M$ elements. If no new tasks/jobs are added, then it is a vector of zeroes, otherwise, it is a vector of zeroes except for the element corresponding to the affected resource which will be equal to one.

K - The number of expected new jobs/tasks to be added to the workload. If no new jobs are added, then K will be equal to one.

The future utilization function will be: $F(U, B, A, P, I, K, T)$ and is denoted by $E(T)$. $E(T)$ is a vector of resource consumption, which determines the relationship between the various resources. Before using regression or any other correlation matrix, the distribution of each variable must be analyzed. The future utilization function may also be written as:

$$E(T) = U + [B \times P + B \times I]^f \times K \tag{17}$$

4.8 Distributed systems

Dinda has created a toolkit called RPS (Resource Prediction System) in order to study and build resource prediction systems [11]. New models to predict with and new components can be created and used by the toolkit. The RPS process involves two steps, creation of an offline evaluation using test data and the creation of an online system to collect and predict data. The methods evaluated in the first step can also be made into a online system. The RPS system have been designed to not be restricted to any specific signal or approach. The system should also easily add new models and components. The components should be able to communicate when placed in different areas of the measured system and should be usable on several platforms. Currently there are four different sensor components in RPS. GetLoadAvg is one of the available sensors, it measures the average run queue length. GetFlowBw measures the available bandwidth between two ip addresses, the WatchTower sensor can on a windows computer access performance counters and proc sensor can on a Linux computer provide access to performance data. As already mentioned above, it is possible to add additional components to the RPS system. The prediction models are used to fit the models to the data, create fitted predictor models, and evaluate the predictions. New models can also be added and compared to the already existing models, that are either linear or non-linear. The linear models in RPS are all implemented using different versions of the following equation:

$$Z_t = \frac{\theta(B)}{\phi(B)(1-B)^d} a_t + \mu \tag{18}$$

Where the Z_t is the measurement sequence, B is the backshift operator, a_t is the white noise sequence, μ is the mean of the recorded values, and $\theta()$ and $\phi()$ are filters with varying coefficients. The MEAN model uses the mean as the predicted value, the LAST model uses the last measured value as the predicted value, the BM predicts the next value to be the average of the previous value.The Auto Regressive model uses the following equation for its predictions:

$$Z_t = \frac{1}{\phi(B)^d} a_t + \mu \tag{19}$$

The Moving Average model uses the following equation for its predictions:

$$Z_t = \theta(B)a_t \tag{20}$$

The Auto Regressive Moving Average uses the following equation for its predictions:

$$Z_t = \frac{\theta(B)}{\phi(B)}a_t + \mu \tag{21}$$

The Auto Regressive Integrated Moving Average model uses the equation (18) for predicting its d value and equation (21) is then applied on the resulting d value. The Auto Regressive Integrated Moving Average model uses equation (18) for its prediction when $0<d<1/2$. The last of the linear models is the WAWELET model where the signal is split up and predicted separately and then combined back into one signal.

The BMED model is one of the nonlinear models and uses the last p values and the average value as its prediction. The NEWTON model uses an interpolating polynomial of p order for its last p values, which then is used for the predictions. However any of the linear models can be used with a wrapper as a nonlinear model.There are a number of wrappers used in RPS to add additional functionality to the different models.

The RPS uses an evaluator implementation that measures the minimum, median, maximum, mean and error predictions are performed for each period of time the system is run. The measured values can then be used to decide if the predictor model have to be refitted. The predictive models are tested using a master/slave parallel evaluation, where the model that is to be tested and a template file containing the test cases have to be provided. The master generates random segments for the model using the template file; the slave evaluates the cases and creates a set of error metrics. The results from the master/slave are then sent to a database where the data can be used to draw conclusions about the prediction model.

To test the toolkit Dinda calculated the run-time costs for fitting the models and creating the predictor, and also for producing 30-steps-ahead predictions. With models that had high initiation costs, the cost for predicting 30-steps-ahead was lower. Two different measurement sequence lengths were measured while being fitted and predicting the steps, with 600 samples and with 2000 samples. Dinda tested the online RPS system for host load with a sample frequency of 1Hz and found it possible to predict 1 to 30 seconds ahead using a AR(16) model, with enough accuracy for the predictions to be useful.

4.9 Neural network vs. regression

Godwin Udo has made a study on comparing a multiple regression approach to an approach based on an ANN, when making a discriminant analysis on which companies are going bankrupt within a two year interval [45]. The multiple regression approaches assumes normality in the variable relationship, which is often not the case when using real-life collected data. Missing values among the sample data is also handled poorly in the multiple regression approach, while the ANN can handle both missing data samples and any variable relationship. The ANN used in the study was developed using a neural net shell called EXPLORENET3000. The study investigated two questions. First, the accuracy at which ANN can predict if a company will go bankrupt within one and two years ahead was studied. Secondly, the ANN and multiple regression were compared. See chapter 3.9 and chapter 3.8 for how ANN and multiple regression works.

In the study, 16 financial ratios was used as variables for the different approaches, with the main categories being capital adequacy, asset quality, liquidity, leverage and profitability. Data about a total of 400 companies was used in the study and of these 200 companies went bankrupt and 200 did not during a two year period. The data was collected from Wall Street Journal, Financial handouts and several other sources. The ANN used had three hidden layers and one output, and was trained using 150 failing and 150 other companies. The networks prediction accuracy was then tested using the remaining 100 data samples. Fitting the multiple regression model was done using the same 300 samples used to train the ANN. Out of the 16 used ratios, 6 were significant at a confident level of 95% for the regression model. The testing of the regression model was done using the same samples used to test the ANN.

The two prediction methods accuracy were compared for the different approaches, one year before and two years before bankruptcy. The result of the comparisons were that ANN were better, or equal to the regression model when the two cases were compared separately and when they were combined. For both approaches, the one-year-before predictions were more accurate than the two-year-before predictions. Udo concluded that ANN provided more accurate predictions, handled change better and were easier to use than the regression model. However, the regression model only used 6 out of the 16 variables available, the ANN approach used all 16 of the variables.

5 Approach

This section describe the process of analyzing the methods from Chapter 4 and finding the most suited method for the OBT departments needs. A model for predicting the required hardware resources for a ECU will also be presented in this section and how the model is to be applied by the OBT department.

5.1 Analysis of methods

The prediction methods described in Chapter 4 were the most relevant methods studied. Some of the methods has similarities in their approaches, although they were conducted in somewhat different areas. Since none of the methods were directly applicable to the TGW, a wide range of areas were studied to form an understanding and identify a suitable approach. The methods described in this thesis is the methods considered to be the most relevant in terms of prediction accuracy and applicability.

One of the limitations for the prediction method is that the predictions should not be made using a continuously running program (see Chapter 1.5) since that approach would require resources from the TGW. Thus, some of the methods were excluded from the possible solutions. The methods that could not be used were HIRES, Two layer queuing, Resource management, Predicting through patterns, and Distributed systems. Neural network vs. regression could also partly be excluded, since neural networks require a running program on the system it is used on. This method was however not excluded since it was deemed a viable option for future development.

One of the aims of the project, described in Chapter 1.3, was that the predictability method should be able to predict when the resources will become full. This aim further supported the decision to exclude HIRES since it makes the system more predictable by distributing the resources, however no actual forecasting is done. An overview of the most interesting methods studied and the reasons for excluding them can be seen in Table 2. The methods with an X in either of the columns named Running or Bottlenecks are excluded due to the reasons explained above.

Table 2: Overview of methods

Method	Running	Bottlenecks
1.HIRES	X	X
2.Two layer queuing	X	
3.Forecasting resource requirements		
4.Resource management	X	
5.Resource usage patterns		
6.Predicting through patterns	X	
7.Computer resource consumption		
8.Distributed systems	X	
9.Neural network vs. regression		

From Table 2 it can be noted that methods number 3, 5, 7 and 9 all were usable for our purposes. These methods will be compared and evaluated in Section 5.2.

5.2 Method selection

All of the methods that were found suitable in chapter 5.1 are more thoroughly examined in this section. The specific method used to predict and the hardware resources that were monitored to make the prediction are to be identified for each of the methods.

In Chapter 4.3, regression analysis was used to predict the level of hardware resource utilization, and eventual bottlenecks encountered were also identified using the regression analysis. The method was tested using data collected from a computer center. The resources used in the test as regression variables were CPU, I/O, and memory.

Chapter 4.5 presents a method that used business prediction models focused on human-generated patterns. The patterns was then used with a SAS tool to generate prediction data. Data about CPU utilization, available memory, network traffic and disk activity on a server was collected and used for predictions. The predictions was then tested by comparing them to predictions made by regression analysis.

Computer resource consumption described in Chapter 4.7 uses a CPE to describe a process of computer selection, configuration design, and to control existing systems. The method measures the current utilization and then calculates the additional utilization using linear regression and adds these two together to decide the future utilization. The resources used in this article were CPU time, memory available, disks activities, and I/O activity.

Chapter 4.9 compares prediction done with multiple regression to predictions made with neural networks. The comparisons were done using data and predictions about companies going bankrupt. Neural networks were found to be better or equal to regression, however the regression analysis required less resources to make its predictions. The resources used for predicting in this study were not interesting for our case.

Patterns found in the TGW are most probably not human-generated and the variables available to monitor are few at the moment. For these reasons regression analysis was the method found most suitable among the methods found. Regression analysis requires less variables for its prediction than neural networks, and does not rely on human-generated patterns like the method presented in Chapter 4.5.

The resources that were monitored or measured for the different methods were CPU utilization, CPU time, I/O activity, network traffic, memory available, and disk activity.

5.3 Our method

The method proposed for predicting resources for the TGW using regression in the following way:

First, information about the resources has to be collected to be able to make any predictions. Information regarding different resources is to be measured by performing different tests cases and using different configurations for the TGW. The test cases are to be re-run each time a program is added, removed or changed, in order to update the information and to receive more data samples for the resources. With a larger number of samples, the predictions made with the regression analysis will become more accurate. The data is to be measured for different test cases, for example heavy load and idle load. For the TGW, the resources identified for monitoring were the CPU utilization, the RAM and Flash memory used, the rates written or read to the memory, the activity on the CAN buses, Ethernet, J1708/J1587 and PPP protocols, the WLAN, and finally the number of processes running on the TGW.

Second, the resource or resources that are to be predicted have to be decided, and what test cases that are interesting to predict. For the regression analysis to be able to predict the variable, resources has to be decided and fitted to the regression model. The predictions can be made for each of the different test cases. However with few samples, predicting resources for a specific test case may provide bad predictions.

Predicting the resource or resources using regression analysis is done in Excel using an add-in which allows regression analysis, or using the statistical language R. Information about the different resources used for the predictions and the predicted resources are then received in a Excel document or in the R program. Description on how to perform the regression is available in Appendix A.1.

Since not all the measured resources affect the predicted resource, some of the resources measured could be removed from the regression. However when removing a resource from the regression, the entire regression analysis has to be redone and new predicted values will be received. The Excel regression analysis add-in and R also provide information about the correlation of the different resources. Information received from the regression can be used to decide what resources affect the prediction and what resources that does not.

After appropriate variable resources have been decided and enough data samples have been collected, a prediction can be done for a specific scenario. The prediction is done by estimating the amount of the variable resources that will be added when making the changes, and the regression then predicts the level of the wanted resource.

When a prediction is done, the prediction is to be compared to a maximum value for the resource in order to identify any bottlenecks that might occur. For example the maximum CPU utilization can be set by the designer or by the hardware.

5.4 Using the method

The method is to be used by the OBT software architects when they plan to implement a new function. The architects can use the method to test how adding different amount of the variable resources affects the prediction values for different test cases. When the architects receive predictions for the different test cases that are acceptable, in terms of the predicted value being implementable on the TGW and within the given limits for the resource, they can proceed. The new function is then to be implemented and verified, the verification is done using different test cases and from running the test cases data about the resources can be measured. The new data can be compared to the predicted values and added to the regression model for more accurate predictions in the future. The process is described in Figure 4 below, and in the Appendix.

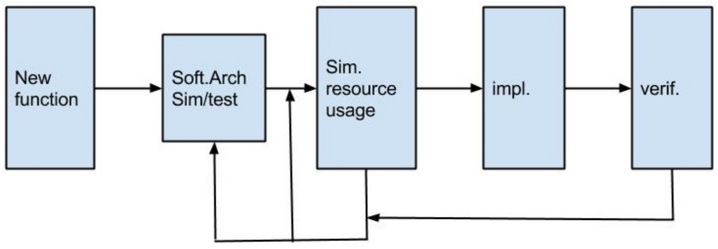


Figure 4: OBT process

6 Results

In this chapter the results from testing the prediction method are described.

6.1 Testing the prediction method

The data used for testing the method can be seen below in Table 3. The data from the table are the usable values from a larger collection of measured data containing 31 releases; the others were omitted due to incompleteness. The resources monitored were the CPU load, which was measured when scrolling in a menu with a frequency of 1 Hz and the GPS running. The RAM memory left, the used Flash, the OS size, and the common software components size. All of the memory resources were measured by studying the memory available on the TGW. The measurements were done on a test rigg running a normal TGW setup. All data with more than 3 decimals have been rounded, to 3 decimals in this section.

Table 3: Historical data samples

Observation	CPU load (%)	RAM left (MB)	Software Flash (KB)	OS size	Softw Component size
1	65	6	10105	2699	1740
2	68	23	10259	2698	1740
3	58	23	10796	2700	2069
4	32	22	10910	2701	2095
5	32	22	11329	2701	2168
6	40	24	9592	2714	1674
7	35	24	9648	2715	1670
8	37	24	9670	2716	1672
9	35	23	9744	2716	1698
10	30	23	10337	2709	1776
11	38	22	10362	2710	1790
12	23	22	10700	2716	1804
13	27	21	10633	2723	1807
14	32	21	10638	3405	1807
15	22	21	10739	2723	1852
16	20	20	10820	3405	1859

Among the values from Table 3, CPU load was the resource that was to be predicted (%), and RAM left (MB), Software Flash (KB), TGW2OS, and TGW2CSWC were the resource variables used for the prediction. The following regression equation was received by using the coefficients received from running the Excel add-in on Table 3:

$$Y = 217.442 + (-1.516)X_1 + (-0.021)X_2 + (-0.0103)X_3 + 0.054X_4 \quad (22)$$

Where Y is the CPU load, 217.442 is the intercept constant, X_1 is the RAM left, X_2 is the Software Flash, X_3 is the TGW2OS, and X_4 is the TGW2CSWC.

Table 4 below shows the comparison of the actual values used from Table 3: CPU load and the predicted values predicted using Equation 22.

Table 4: Predicted values and residual

Observation	Predicted CPU load (%)	Residuals	Fault (%)
1	62.699	2301	3.540
2	33.707	34.293	50.431
3	40.265	17.735	30.578
4	40.786	-8.786	-27.457
5	35.945	-3.945	-12.328
6	42.460	-2.460	-6.150
7	41.056	-6.056	-17.302
8	40.692	-3.692	-9.977
9	42.064	-7.064	-20.182
10	33.909	-3.909	-13.031
11	35.650	2.350	6.185
12	29.245	-6.245	-27.152
13	32.260	-5.260	-19.480
14	25.137	6.863	21.448
15	32.475	-10.475	-47.615
16	25.651	-5.651	-28.256

The Residuals are the difference between the observed value and the predicted value. A negative residual value indicate that the prediction is too high and a positive value indicated that its too low. The Fault (%) is the size of the residual compared to the observed value.

6.2 Test cases

Two test cases were created for the measuring the resource usage of the TGW under different circumstances. One test case for when the TGW is running under heavy load, and the other test case is for idle load. No usable data was gathered, however; the reasons are discussed in Chapter 7.3.2.

7 Discussion

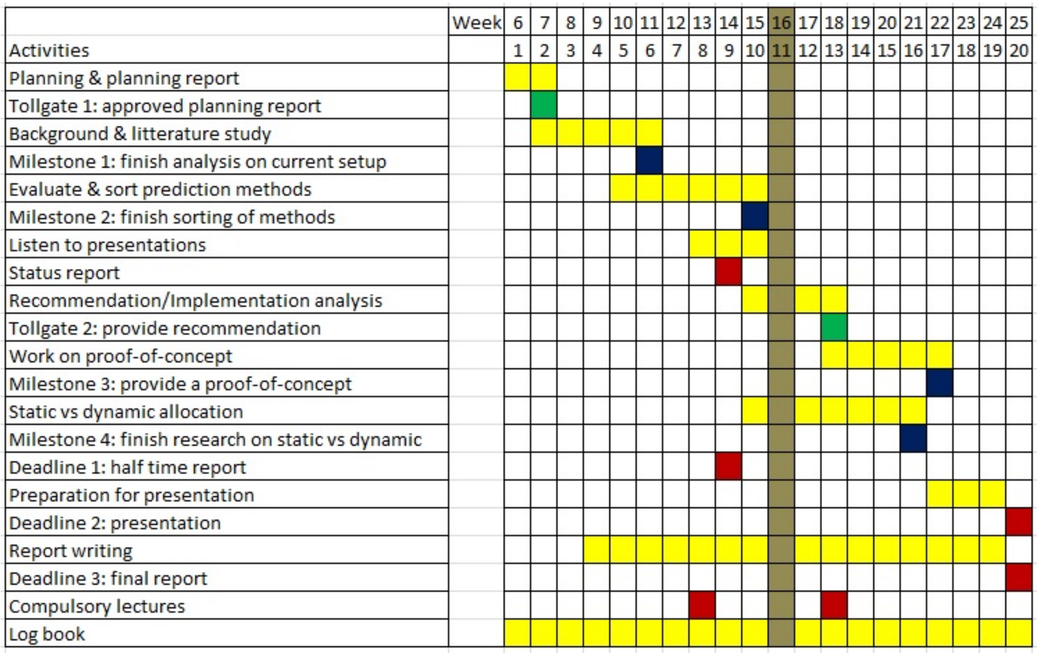
In this sections the execution of the project, results received from testing, and the prediction method are discussed and reflected upon.

7.1 Review of plan

In chapter 1 the purpose, goal and limitations of this project was stated. In chapter 2 a procedure for how the project should be carried out was planned together with a timeplan. Did it go according to plan and was the timeplan followed?

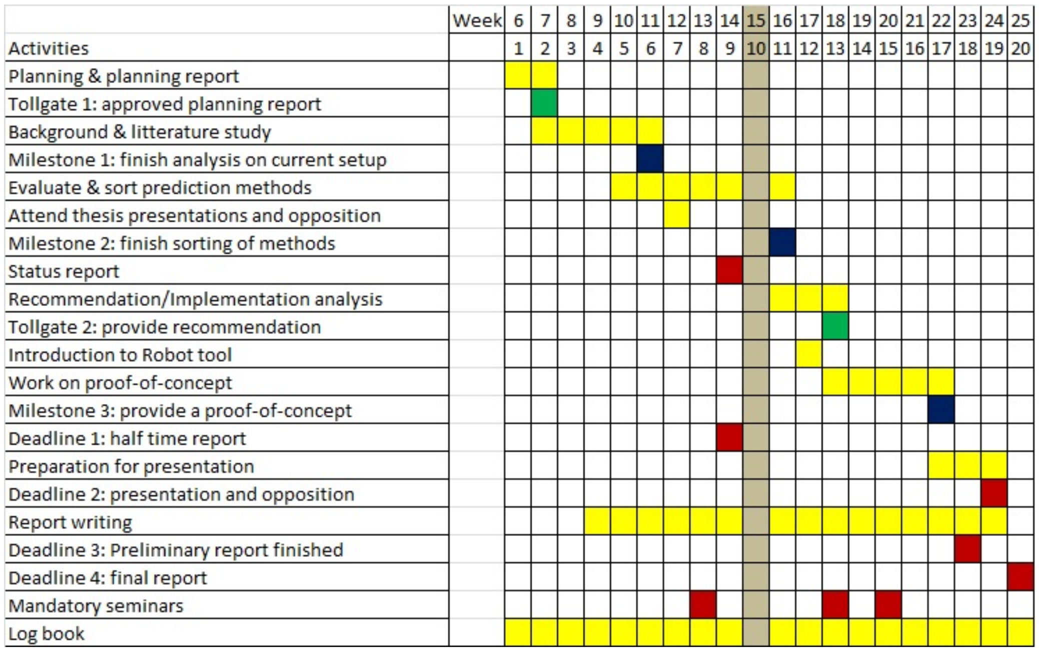
The timeplan and work process was followed according to the original plan until the half-time had passed. At that point a review of the original plan was made which resulted in some minor changes. The milestone for the sorting of methods was moved since the date for the break was moved one week earlier. Also, the dates for the hand-in of the report and the presentation was adjusted since the exact dates was unknown before.

The major change done was a couple of weeks after the half-time review. After discussion with our supervisor at Volvo the part of the project involving static vs dynamic memory allocation was removed. This was done because the OBT department at Volvo decided that they did not want an analysis of memory allocation methods, but wanted a technical solution. This would require significant more work than performing a GAP-analysis of the current allocation method and possible methods. Since it would require more time than we had, that part was removed. Aside from that, we managed to follow the original timeplan, Figure 5. The final timeplan can be seen in Figure 6.



	Week	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Activities		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Planning & planning report		🟨	🟨									▨									
Tollgate 1: approved planning report			🟩									▨									
Background & litterature study			🟨	🟨	🟨	🟨	🟨					▨									
Milestone 1: finish analysis on current setup							🟦					▨									
Evaluate & sort prediction methods						🟨	🟨	🟨	🟨	🟨	🟨	▨									
Milestone 2: finish sorting of methods											🟦	▨									
Listen to presentations									🟨	🟨	🟨	▨									
Status report										🟥		▨									
Recommendation/Implementation analysis											🟨	▨	🟨	🟨							
Tollgate 2: provide recommendation												▨		🟩							
Work on proof-of-concept												▨		🟨	🟨	🟨	🟨	🟨			
Milestone 3: provide a proof-of-concept												▨						🟦			
Static vs dynamic allocation											🟨	▨	🟨	🟨	🟨	🟨	🟨				
Milestone 4: finish research on static vs dynamic												▨					🟦				
Deadline 1: half time report										🟥		▨									
Preparation for presentation												▨						🟨	🟨	🟨	
Deadline 2: presentation												▨									🟥
Report writing					🟨	🟨	🟨	🟨	🟨	🟨	🟨	▨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
Deadline 3: final report												▨									🟥
Compulsory lectures									🟥			▨		🟥							
Log book		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	▨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

Figure 5: The initial timeplan for the project.



	Week	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Activities		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Planning & planning report		🟨	🟨																		
Tollgate 1: approved planning report			🟩																		
Background & litterature study			🟨	🟨	🟨	🟨	🟨														
Milestone 1: finish analysis on current setup							🟦														
Evaluate & sort prediction methods						🟨	🟨	🟨	🟨	🟨		🟨									
Attend thesis presentations and opposition								🟨													
Milestone 2: finish sorting of methods												🟦									
Status report										🟥											
Recommendation/Implementation analysis												🟨	🟨	🟨							
Tollgate 2: provide recommendation														🟩							
Introduction to Robot tool													🟨								
Work on proof-of-concept														🟨	🟨	🟨	🟨	🟨			
Milestone 3: provide a proof-of-concept																		🟦			
Deadline 1: half time report										🟥											
Preparation for presentation																		🟨	🟨	🟨	
Deadline 2: presentation and opposition																				🟥	
Report writing					🟨	🟨	🟨	🟨	🟨	🟨		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
Deadline 3: Preliminary report finished																			🟥		
Deadline 4: final report																					🟥
Mandatory seminars									🟥					🟥		🟥					
Log book		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

Figure 6: The actual timeplan for the project.

Initially, we identified the components, devices, and protocols in Chapter 3 as key elements to study when performing the prediction. Even though the expected deliverable was a simple proof-of-concept, our plan was to try and implement the method and identify what affects the CPU load. However, we had to shift focus back to a simple proof-of-concept since we reached the conclusion that implementing the method with those elements would take more time than we had. Even thought we do not really address the elements of Chapter 3 in our approach, we started researching them and still believe that studying them is important for future work.

Furthermore, the work on the proof of concept was more complicated than we initially thought. Although we did manage to stay within our timeframe, we did not manage to get enough data to be able to perform a reliable prediction. One reason for that was the time aspect since it gathering the data required more time than we had planned for. Another reason for that was a lack of documentation at Volvo OBT regarding the programming language used in a tool called Robot, which meant that creating and testing test cases took a little more time than it should have.

Furthermore, a problem occurred with the programming tool R. Initially, the goal was to perform the prediction through R, however problems occurred during the installation process. The packages that are installed with the program could not be loaded. The reason was that R was installed on the network drive which resulted in problems with the file path. We tried to install it on the local drive instead, however the program could not be installed at all. During the installation an error emerged regarding some of the system files that cannot install, and we did not manage to find the reason why. R was installed on our private computers instead and a script was created to fetch data from a Microsoft Excel sheet and then perform the regression. We could however not test this on Volvo's computers. Eventually, we performed the regression analysis entirely in Microsoft Excel, which is a bit more tedious than running it through R.

Aside from that some of the time we spent in the beginning of the project could have been spent differently. Initially, we had no restrictions and was looking into all kind of prediction methods. However, as time progressed and discussions were held with our supervisor at Volvo some limitations had to be made. The conditions we were left with was that the prediction method should be for their current product which meant that we had to weed out methods that required redesign of the product's system. Furthermore, it was decided that the prediction method should not be a running program on the system but outside it instead. This means that the weeks we spent researching their current system could have been shortened and more time could have been spent on the methods or the proof of concept.

7.2 Choice of method

The choice of using regression analysis for predicting the hardware resources for the TGW was made because the OBT department could implement it on the TGW. Furthermore, regression analysis was used among many of the articles and methods studied. Some of the studied material covered areas that may not seem as relevant for our work, and in those cases, what we were looking for is how diverse their methods were. Regression was used in different areas and can be applicable to more than one specific resource. It is possible to add additional resources when predicting, as long as the added resources have the same amount of data samples as the existing resource samples.

The resources, which covers what we want to predict and what we use to predict, that are suitable for the process described in chapter 5.3 were chosen for a range of different reasons. The resources identified in Chapter 5.2 were chosen after studying and evaluating the articles and the other resources were selected after interviews and discussions with personnel at OBT.

Regression analysis was the only suitable method that was found during our literature study for predicting resources without having a process running continuously on the TGW.

Business prediction models focused on human-generated patterns were one of the options that were studied as a possible prediction method. However, due to the fact that this method used patterns and finding patterns for the TGW could prove difficult, and the resources predicted were limited, this model was eliminated as a possible prediction method.

Neural networks was something that we looked into but quickly decided would not be applicable. One reason is that it would require too much resources as a running program. Furthermore, training a neural system takes a significant amount of time (can take up to six months and more) and it is harder to interpret the output values compared to regression analysis. The interpretation is more difficult because it is hard to know what the network actual does when it processes the information. The amount of work and time required to get a neural network up and running is most likely not worth the end result since regression analysis can achieve similar accuracy with less variables. It is also easier to change variables in regression analysis. Finally, the amount of hardware resources required to sustain a neural network can be used for other work or simply scaled down in order to lower the cost.

7.3 Method test result

In this section the results received from testing the method and making test cases will be discussed.

7.3.1 Testing the prediction model

When studying the results (Table 4) received from using regression analysis on the values from Table 3 it can be noted that the faults when predicting the CPU load varied from 50% to 3.5%. While the predictions performed were not very accurate, they still prove that predictions using multiple regression is possible and will work when predicting the CPU load for the TGW. Since the residual for many of the predictions were high. Considering that only one regression analysis was done and no new data samples were added it is hard to estimate to what degree adding more samples will improve the predictions.

The results received could have been better if the resources identified for monitoring would have had more samples available. The accuracy of the predictions suffers from the low amount of data samples available and more samples would have improved the accuracy. Furthermore, the data samples from Table 3 did not include data from the most recent versions of the TGW releases and there is a gap of missing data between some releases since they were not measured for unknown reasons. The missing data would have contributed with more samples which could have resulted in a more accurate prediction. The data samples used for this test covered only few of the identified resources and their impact on the predicted value is hard to determine since the samples and resources monitored are few in numbers. Thus, identifying which resources affects the prediction the least and what resources to add to the prediction algorithm was impossible.

7.3.2 Test cases

As mentioned in Chapter 6.2, two test cases was created in order to gather data for the predictions. However, the simulation did not work properly. Aside from the documentation problem described in chapter 7.1, there were some technical problems with the testing environment. The problems revolved around some of the protocols and communication between the tools. We were able to identify the problems, however we could not solve them.

A lot of the resources identified in Chapter 5.3 were not currently measurable from the test cases, which means that a method for measuring these resources will have to be implemented. The test cases will also have to be run for several of the older versions of the TGW software, in order to receive enough samples for the predictions.

7.4 Prediction

Without testing all of the hardware resources for the TGW, it is hard to know if the resources chosen as variables are best for making the wanted predictions. Some additional variables and factors, external and internal, that could have been studied are the temperature of the TGW and the power consumption of the TGW, since they most likely affect the processors performance. Other resources could also be affected by external factors. The resources used for the predictions have to be considered carefully and depending on the resource that is to be predicted the resources used for the prediction have to be adapted.

When adding new variables that are used for the predictions to the already existing samples of other variables, the older samples cannot be used since the regression requires the same amount of samples for each of its variables. This means that the history currently used for the prediction becomes obsolete and must be reseted. An alternative method would be to make a "reverse" prediction and predict backwards to produce reasonable values for the new parameter. This would have to be done before making the actual prediction for the wanted resource, to be able to use all of the older sample values. A problem however with predicting the missing values is deciding if the predicted values are to be updated each time new samples are added or if this is to be done less often. The impact on updating the missing values each time new samples are added needs to be investigated before relying on the results from predictions using these samples. Due to the fact that the predictions become more accurate with larger amount of data samples, the prediction accuracy at the start of the predictions will be low. The accuracy will, as stated above, increase with time as more samples are measured.

The test cases provided a way to measure the values for different scenarios that may prove interesting for the predictions. For example predicting for a case with heavy load provides predictions where the values are higher than usual and can be seen as a highest possible value for the TGW.

Something that also will matter when measuring and predicting for different test cases is the configuration of the TGW. Making predictions for both different test cases and different configurations of the TGW will probably be required for the method to make good predictions.

A problem with the method is that whenever the hardware is changed the variables needs to be reevaluated and new samples must be collected. The prediction accuracy will be lower due to the smaller amount of samples available for the variables when measuring new samples. However, measuring new samples also provides an opportunity to introduce new test cases and new resources. The resources added could be new due to the changes made in hardware or software architecture or old resources that have not been measured previously.

Estimating the required resources before making the prediction is another potential problem. Before starting the implementation it is difficult to know or estimate the amount of resources that will be used for the finished function which can make the predictions less accurate. Previously measured data can be used as a guideline when making estimations for new functions. Documentation regarding the changes and estimations made must be done properly.

During our research a patent covering the use of regression in computer systems was encountered. The patent ”Predicting hardware usage in a computing system US 8417811 B1” was, according to us, unclear in what it exactly covered [23]. Our interpretation is that the patent covers using one or more regression analysis in a computing system. Our regression analysis will be carried out in the testing phase and outside the products resources. We are not sure if our approach is outside the range of the patent and thus, it is our recommendation to thoroughly investigate this patent if further work and implementation of the method described in thesis will be carried out.

7.5 Expanding the method

This project has been focusing on making predictions for the TGW alone. If the predictions using regression analysis were to be expanded to a larger system, e.g. an entire truck or car, some considerations must be made. In our case, the TGW has a high growth rate but that might not be the case for other ECU's. ECU's with lower growth rate might be more difficult to predict accurately since measuring samples for the resources will be done less regularly thus giving fewer samples for the predictions. Furthermore, the ECU's impact on each other must be considered, e.g. which ECU's communicate with each other, and what they do. The larger the system, and the more ECU's that are involved, it becomes more and more complicated to use regression and other alternatives might be better.

First of all, our approach to regression where the calculations are being carried out outside the system will not be viable. Our suggestion is to use software monitoring instead in order to gather the data rather than extract it manually and store it in a file. A neural network might be more viable in a larger system even though it has its flaws, as discussed in Section 7.2. A drawback that might occur when running the prediction methods on a system is that the predictions will differ for every different vehicle since the vehicles has a different path and usage patterns. A workaround could be to only use the prediction during testing as well, but instead of simulations, like our method, the method is applied when test driving vehicles.

8 Conclusion

The purpose of this project was to investigate and evaluate prediction methods for hardware resource usage for the Telematics gateway (TGW). 60 articles was evaluated and sorted down to nine. The remaining articles were then studied more thoroughly and regression analysis was chosen for predicting the resource usage.

Throughout the project, limitations were added which did not change the scope of the work but affected the criteria for the prediction method. Furthermore, one of the goals for the project, the gap analysis for dynamic vs static memory allocation, was removed and focus was only put on the prediction method. In the end, a method was chosen and tested. However, the tests were not optimal since the necessary data could not be gathered properly from the TGW. Instead, the prediction was tested in general with the data that was accessible in order to determine its validity. The proposed method firstly gathers resource data. Secondly, it performs a regression analysis to make predictions. The predictions can then indicate whether the current TGW resources will be able to handle it or not.

In the end, this thesis work did manage to achieve the stated goals and deliver a recommendation for a prediction method as well as a simple proof of concept.

8.1 Future work

Measuring the TGW's resources must be done more regularly and for a larger range set of data. In other words, history for the TGW's resources must be gathered over time and used for the predictions. More data means more accurate predictions. The amount of test cases which are used to measure the resource usage should be increased for more data. Running existing test cases can be used in order to build a history database.

An option for measuring the hardware resources could be to add a running program on the TGW in order to receive more data samples. This is however not an option for the current version of the TGW since this will require resources from the already limited resources.

If regression proves to make predictions which are unusable by the OBT department, implementing a neural network could be a option for future TGW versions. However, this requires a program which is run constantly for a long period of time and needs more input variables than regression in order to make predictions. The prediction process for the neural network is also harder to interpret and understand than regression. Furthermore, it is difficult to estimate how the network will be able to handle large system changes.

9 References

[1] CAN Specification 2.0, Part B. [ebook] Erlangen: CAN in Automation. Available at: `http://www.can-cia.org/index.php?id=downloads-other-specs`, 2012. [Accessed 27 April 2015].

[2] Niv Ahituv and Magid Igbaria. A model for predicting and evaluating computer resource consumption, 1988.

[3] Andale. Statistics how to. `http://www.statisticshowto.com/what-is-the-pearson-correlation-coefficient/`, 2012. [Accessed 30 April 2015].

[4] Vangie Beal. The 7 Layers of the OSI Model. `http://www.webopedia.com/quick_ref/OSI_Layers.asp`, 1999. Updated 22 April 2015, [Accessed 3 June 15].

[5] Vangie Beal. What is Random Access Memory (RAM)? `http://www.webopedia.com/TERM/R/RAM.html`, 2015. [Accessed 20 April 2015].

[6] Vangie Beal. The 7 Layers of the OSI model. `http://www.webopedia.com/quick_ref/OSI_Layers.asp/`, updated 2015. [Accessed 20 April 2015].

[7] Daniel E. Capano. Wi-Fi and the OSI model — Control Engineering. `http://www.controleng.com/single-article/wi-fi-and-the-osi-model/8b71b0494b6b7fd5291856d02e104eb4.html`, 2014. [Accessed 20 April 2015].

[8] J Carter. Telematics: what you need to know. `http://www.techradar.com/news/car-tech/telematics-what-you-need-to-know-1087104`, 2012. [Accessed 5 February 2015].

[9] N Chandrashekara. Electronic Control Unit(ECU). [pdf document] ETAS. Available at: `http://www.etas.com/data/group_subsidiaries_india/20140121_ETAS_Webinar_ECU_Basics.pdf`, 2014. [Accessed 30 April 2015].

[10] Karin Denti. Predictable HW resource usage in embedded systems. [pdf document] Volvo Group Trucks technology, 2014.

[11] P. A. Dinda. Design, implementation, and performance of an extensible toolkit for resource prediction in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 17(2):160–173, 2006.

[12] Subhasri Duttagupta, Rupinder Virk, and Manoj Nambiar. Predicting performance in the presence of software and hardware resource bottlenecks. pages 542–549. IEEE, 2014.

[13] Explorable.com. Multiple Regression Analysis. `https://explorable.com/multiple-regression-analysis/`, 2009. [Accessed 17 April 2015].

[14] John Fuller. Why are the sales of electronics for your car expected to rise 12 percent this year? `http://auto.howstuffworks.com/under-the-hood/trends-innovations/car-electronics-increase.htm`, 2008. [Accessed 13 February 2015].

[15] Gartner.com. Telematics. `http://www.gartner.com/it-glossary/telematics`, 2015. [Accessed 21 March 2015].

[16] Patrick E. George. Have new technologies made cars less safe? `http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/new-technologies-cars-less-safe1.htm`, 2010. [Accessed 13 May 15].

[17] Volvo Group Global. Volvo Group Organization. `http://www.volvogroup.com/group/global/en-gb/volvo%20group/our%20companies/Pages/business_structure.aspx`, 2014. [Accessed 28 May 15].

[18] Embedded Systems Guide. Electronic Control Unit (ECU). `http://www.embedded-systems-portal.com/CTB/Electronic_Control_Unit,1008.html`, 2015. [Accessed 30 April 2015].

[19] Emad Isaac. Telematics: linking the future now. `http://www.oemoffhighway.com/article/10181908/telematics-linking-the-future-now`, 2010. [Accessed 5 February 2015].

[20] Cory Janssen. Dynamic Random Access Memory (DRAM). `http://www.techopedia.com/definition/2770/dynamic-random-access-memory-dram`, 2015. [Accessed 15 March 2015].

[21] Cory Janssen. Static Random Access Memory (SRAM). `http://www.techopedia.com/definition/2814/static-random-access-memory-sram`, 2015. [Accessed 15 March 2015].

[22] Cory Janssen. What is the Point-to-Point Protocol (PPP)? `http://www.techopedia.com/definition/25315/point-to-point-protocol-ppp`, 2015. [Accessed 20 April 2015].

[23] J.A. Jenkins and T.M. Sehn. Predicting hardware usage in a computing system, April 9 2013. US Patent 8,417,811.

[24] Srishty Jindal and Priyanka Grover. Two queue based round robin scheduling algorithm for cpu scheduling. *International Journal of Computer Applications*, 105(5), 2014.

[25] Kvaser. Introduction to SAE J1587 - Kvaser. `http://www.kvaser.com/about-can/can-standards/j1587/`, 2015. [Accessed 21 April 2015].

[26] Kvaser. Introduction to SAE J1708 - Kvaser. `http://www.kvaser.com/about-can/can-standards/j1708/`, 2015. [Accessed 21 April 2015].

[27] David M Lane. Online statistics education: An interactive multimedia course of study. `http://onlinestatbook.com/2/regression/intro.html`. [Accessed 30 April 2015].

[28] David M Lane. Online statistics education: An interactive multimedia course of study. `http://onlinestatbook.com/2/regression/accuracy.html`. [Accessed 30 April 2015].

[29] MarineStatsLectures. Multiple Linear Regression in R (R tutorial 5.3). [Online Video] Available at: `https://www.youtube.com/user/marinstatlectures/videos`. [Accessed 04 May 2015].

[30] Bradley Mitchell. WLAN - Wireless Local Area Network Described. `http://compnetworking.about.com/cs/wirelessproducts/g/bldef_wlan.htm`, 2015. [Accessed 20 April 2015].

[31] Dipti Ranjan Mohanty. Telecom-Network Technology: OSI Reference MODEL & Protocols Model- TCP/IP, photograph. `http://teleco-network.blogspot.se/2012/11/osi-reference-model-protocols-model.html`, 2012. [Accessed 11 May 15].

[32] Gabriel Parmer and R. West. Hires: A system for predictable hierarchical resource management. pages 180–190. IEEE, 2011.

[33] Nick Pidgeon. How Ethernet works. `http://computer.howstuffworks.com/ethernet.htm/`, 2000. [Accessed 17 April 2015].

[34] Michael Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, New York, 2012.

[35] Ian Poole. What is Flash Memory? `http://www.radio-electronics.com/info/data/semicond/memory/what-is-flash-memory-basics-tutorial.php`, 2015. [Accessed 30 April 2015].

[36] Robert Prime. Telematics History And Future Predictions. `http://www.telematics.com/telematics-history-future-predictions/`, 2013. [Accessed 21 March 2015].

[37] Daniel Shiffman. The Nature of Code. `http://natureofcode.com/book/chapter-10-neural-networks//`, 2012. [Accessed 27 April 2015].

[38] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. John Wiley & Sons Inc, Hoboken, N.J, 2009.

[39] W Simpson. The Point-to-Point Protocol (PPP). `http://tools.ietf.org/html/rfc1661`, 1994. [Accessed 20 April 2015].

[40] Zilla Sinuany-Stern and Hadar Yelin. Forecasting hardware resource requirements. *Computers and Operations Research*, 20(5):477–484, 1993.

[41] Stattrek.com. Statistics and probability dictionary. `http://stattrek.com/statistics/correlation.aspx?Tutorial=AP`, 2015. [Accessed 30 April 2015].

[42] Jian Tan, P. Dube, Xiaoqiao Meng, and Li Zhang. Exploiting resource usage patterns for better utilization prediction. pages 14–19. IEEE, 2011.

[43] Jeff Tyson. How flash memory works. http://computer.howstuffworks.com/flash-memory2.htm/, 2000. [Accessed 30 April 2015].

[44] Ryoma Uchida, Hiroto Horino, and Ren Ohmura. Improving fault tolerance of wearable wearable sensor-based activity recognition techniques. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 633–644, New York, NY, USA, 2013. ACM.

[45] Godwin Udo. Neural network performance on the bankruptcy classification problem. *Computers & Industrial Engineering*, 25(1):377–380, 1993.

[46] Volvo. System specification Telematic Gateway TGW2X. [pdf document] Volvo Group Trucks technology, 2014.

[47] Wikipedia. Carrier sense multiple access with collision detection. http://en.wikipedia.org/wiki/Carrier_sense_multiple_access_with_collision_detection, 2015. [Accessed 1 June 15].

[48] Wikipedia. SAS (software). http://en.wikipedia.org/wiki/SAS_%28software%29, 2015. [Accessed 17 May 15].

[49] Henrik Wärmlind. (2015) Interview by Einar Nilsson and Mohammed Elghoz, 1 June 2015.

[50] Chi Yau. R tutorial. http://www.r-tutor.com/elementary-statistics/multiple-linear-regression/multiple-coefficient-determination, 2015. [Accessed 1 May 2015].

[51] Daniel Yoas and Greg Simco. Resource utilization prediction: a proposal for information technology research. pages 25–30. ACM, 2012.

[52] Jeff Zagoudis. Telematics Puts Managers In the Driver's Seat. http://www.constructionequipment.com/

`telematics-puts-managers-driver%E2%80%99s-seat`, 2011. [Accessed 21 March 2015].

[53] Qiao Zhang and Chuanchang Liu. Workload prediction in load balancing and resource management system. *Information Technology Journal*, 12(21):6086–6089, 2013.

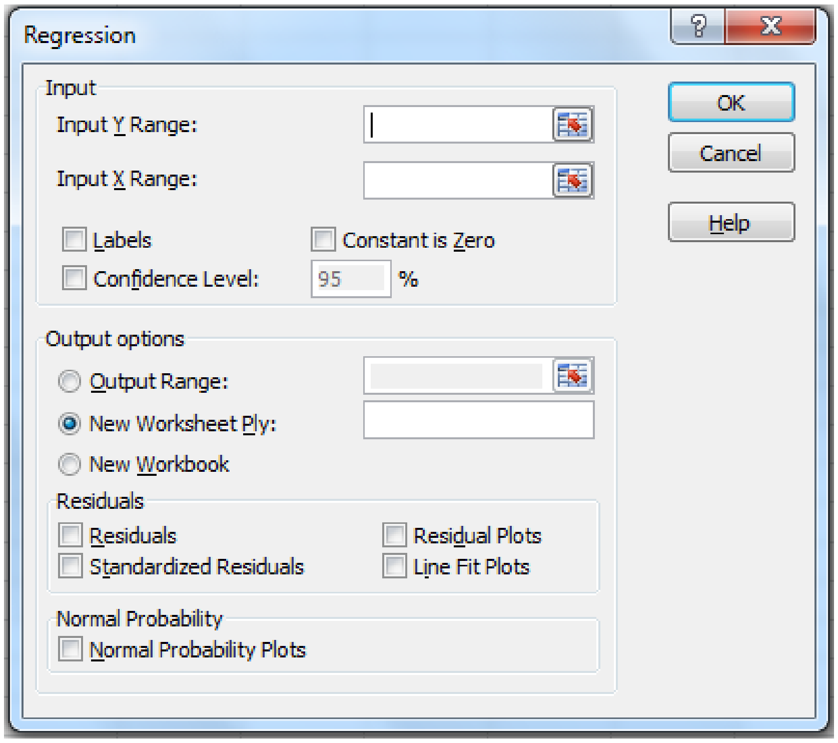
A Appendix

A.1 Performing the regression

This section describes how the regression analysis can be performed using the Excel add-in or the statistical language R.

A.1.1 Analysis Toolpak (Excel add-in)

The Analysis Toolpak add-in is included when installing Microsoft Excel and can be managed from the add-in menu in options. The variables used for the predictions can not have any blank spaces among the values. While in Excel, the regression can be accessed from the data tab and data analysis, the window seen in Figure 7.



Regression

Input
Input Y Range:
Input X Range:
☐ Labels ☐ Constant is Zero
☐ Confidence Level: 95 %

Output options
○ Output Range:
● New Worksheet Ply:
○ New Workbook

Residuals
☐ Residuals ☐ Residual Plots
☐ Standardized Residuals ☐ Line Fit Plots

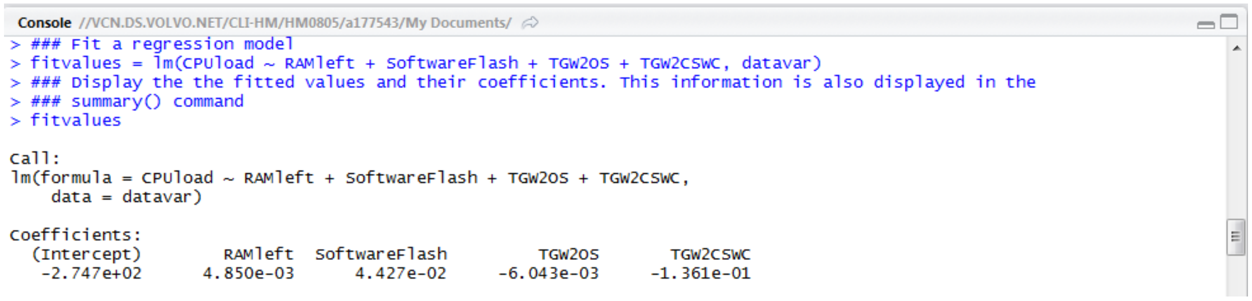
Normal Probability
☐ Normal Probability Plots

Figure 7: Regression window Excel

The observed data for the variable to be predicted is chosen as the input Y range and the data for the variables used for the predictions is chosen as input X range. The user also has to decide where the output of the regression should be sent, to a new page, to an existing page or to a new project. The final choice the user can make is what residuals and probabilities to include for the results. To make the prediction, the Y and X values have to be chosen and they need the same number of data samples. The constants for the regression and additional information about the regression are then obtained by clicking the OK button. The results generated are the predictions for the sample data, regression statistics, analysis of variance (ANOVA), the regression constants, and if chosen residual plots will be created at the chosen location. The constants can then be used for predicting the the resources, by placing them in the regression Equation 7 from Chapter 3.8.5.

A.1.2 R (Statistical and graphical programming language)

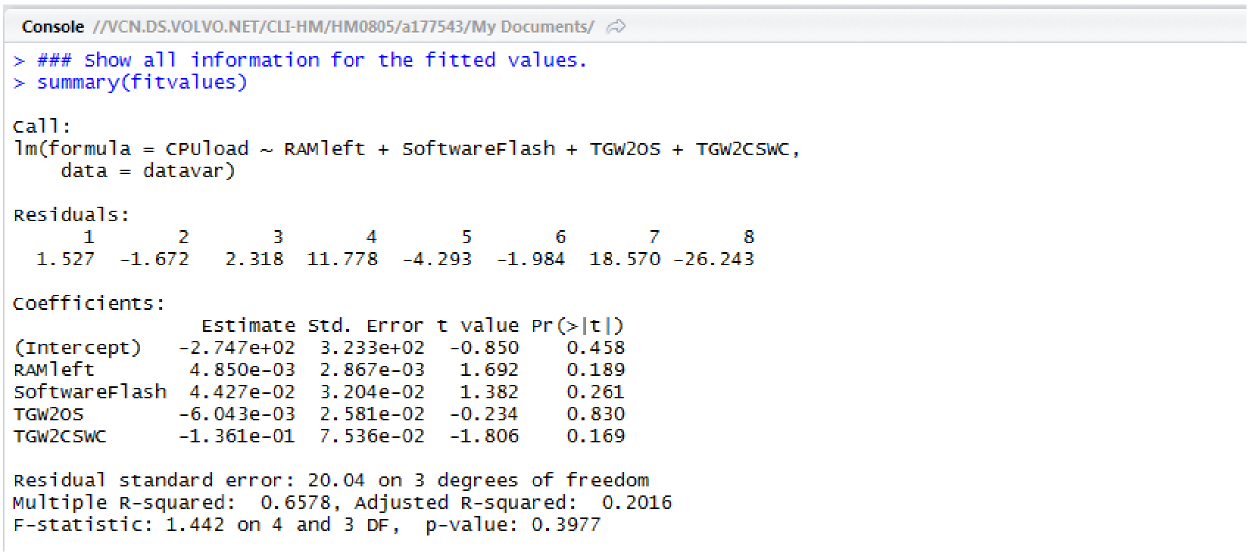
An alternative, and more efficient, way to perform the regression and analyze the data is to use programming language R with the Rstudio program.



```
Console //VCN.DS.VOLVO.NET/CLI-HM/HM0805/a177543/My Documents/
> ### Fit a regression model
> fitvalues = lm(CPUload ~ RAMleft + SoftwareFlash + TGW2OS + TGW2CSWC, datavar)
> ### Display the the fitted values and their coefficients. This information is also displayed in the
> ### summary() command
> fitvalues

Call:
lm(formula = CPUload ~ RAMleft + SoftwareFlash + TGW2OS + TGW2CSWC,
    data = datavar)

Coefficients:
  (Intercept)        RAMleft  SoftwareFlash         TGW2OS       TGW2CSWC
   -2.747e+02      4.850e-03      4.427e-02     -6.043e-03     -1.361e-01
```

Figure 8: Fitting a model and the output produced.

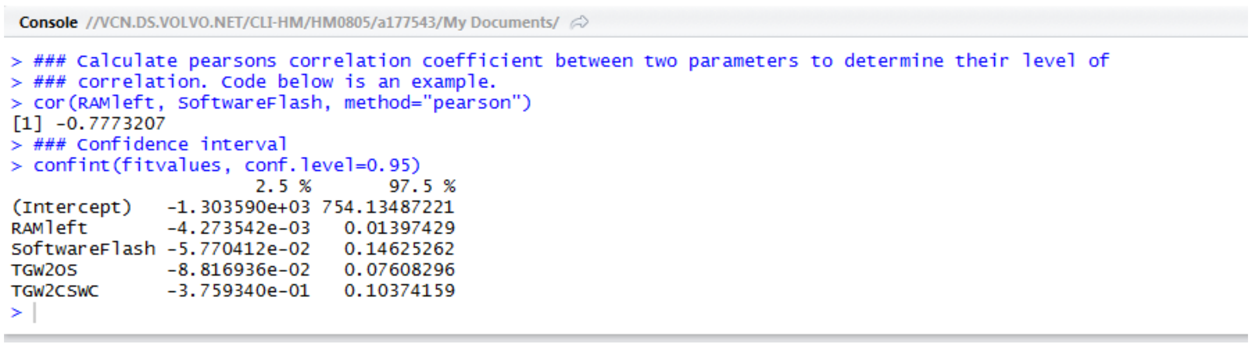
The `lm()` command is used for fitting a linear model, as seen in figure 8, which results in the coefficients for the variables [29].


Console //VCN.DS.VOLVO.NET/CLI-HM/HM0805/a177543/My Documents/

```
> ### Show all information for the fitted values.
> summary(fitvalues)

Call:
lm(formula = CPUload ~ RAMleft + SoftwareFlash + TGW2OS + TGW2CSWC,
    data = datavar)

Residuals:
      1       2       3       4       5       6       7       8
  1.527  -1.672   2.318  11.778  -4.293  -1.984  18.570 -26.243

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.747e+02  3.233e+02  -0.850    0.458
RAMleft        4.850e-03  2.867e-03   1.692    0.189
SoftwareFlash  4.427e-02  3.204e-02   1.382    0.261
TGW2OS        -6.043e-03  2.581e-02  -0.234    0.830
TGW2CSWC      -1.361e-01  7.536e-02  -1.806    0.169

Residual standard error: 20.04 on 3 degrees of freedom
Multiple R-squared:  0.6578, Adjusted R-squared:  0.2016
F-statistic: 1.442 on 4 and 3 DF,  p-value: 0.3977
```

Figure 9: Output from the summary() command

By using the `summary()` command after fitting the model, different values are received [29]. The multiple R-squared data achieved in Figure 9 is an approximation of how much variation in percent for the CPU load that can be explained by the variables RAMleft, SoftwareFlash, TGW2OS, and TGW2CSWC. The F-statistic tests the null hypothesis, which means that it tests whether the independent variables collectively have an effect on the dependent variable. The residual standard error is used to give an approximation of how much difference there is between the observed values and predicted values for the dependent variable that we want to predict. The intercept data is the estimated mean for the dependent variable when all variables used for the prediction are 0. Below the intercept coefficient is the coefficients for the variables used which represents the effect of RAMleft on CPUload adjusting for the other variables.

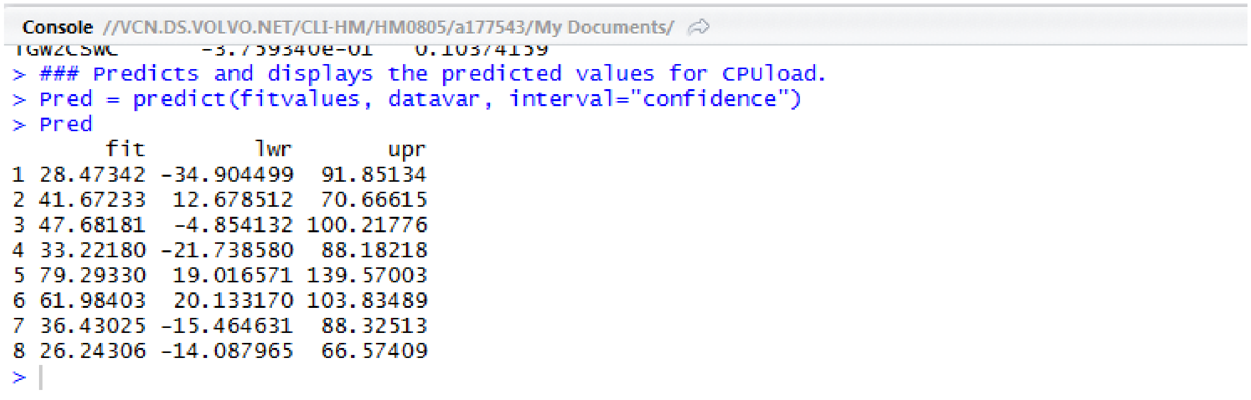


Console //VCN.DS.VOLVO.NET/CLI-HM/HM0805/a177543/My Documents/

```
> ### Calculate pearsons correlation coefficient between two parameters to determine their level of
> ### correlation. Code below is an example.
> cor(RAMleft, SoftwareFlash, method="pearson")
[1] -0.7773207
> ### Confidence interval
> confint(fitvalues, conf.level=0.95)
                      2.5 %       97.5 %
(Intercept)   -1.303590e+03 754.13487221
RAMleft       -4.273542e-03   0.01397429
SoftwareFlash -5.770412e-02   0.14625262
TGW2OS        -8.816936e-02   0.07608296
TGW2CSWC      -3.759340e-01   0.10374159
> 
```

Figure 10: The commands for correlation analysis and confidence interval

The `cor()` command gives the correlation between two variables [29]. In other words, the value indicates how much the values are bounded together, with the value 1 meaning that they are completely bounded together. The `confint()` command creates a confidence interval which gives an 95% that the slope are in between the given values in Figure 10.



Console //VCN.DS.VOLVO.NET/CLI-HM/HM0805/a177543/My Documents/

```
TGW2CSWC      -3.759340e-01   0.10374159
> ### Predicts and displays the predicted values for CPUload.
> Pred = predict(fitvalues, datavar, interval="confidence")
> Pred
       fit        lwr       upr
1 28.47342 -34.904499  91.85134
2 41.67233  12.678512  70.66615
3 47.68181  -4.854132 100.21776
4 33.22180 -21.738580  88.18218
5 79.29330  19.016571 139.57003
6 61.98403  20.133170 103.83489
7 36.43025 -15.464631  88.32513
8 26.24306 -14.087965  66.57409
> 
```

Figure 11: Using the `predict()` command to predict the values for CPUload

Finally, in Figure 11, we can see how the `predict()` command is used to make a prediction for the CPUload using the previously entered data as well as including an interval the accuracy. Important to note is that these commands have been tested and explained with the use of made up values and measured values from the TGW.