# CASE 1: Using SIR--models to Estimate the Bonus Effects of Influenza Mass Vaccination

## 1 Introduction

In the early year of 2009, the World Health Organization(WHO) raised an alarm that a new strain of influenza had broken out in Mexico. Later that year, the outbreak was classified as a pandemic by WHO as the virus spread across the globe (Chan 2009). Different countries adopted different policies for dealing with the pandemic in order to minimize its impact on society. One of the policies that were used was mass vaccination, the distribution of vaccine to large parts of the population (The Guardian, 2009). How the vaccinations affected the spread of infection the following season is unknown.

### 1.1 Background

In systems theory, there is a special approach for understanding complex systems' behavior over time: System Dynamics (SD). SD is a mathematical modelling technique, with a high level of abstraction. The technique aids in analysing, understanding, and discussing complex situations and problems. SD models take a holistic view of a system, that is, the system is viewed as a whole and each little factor can have an impact on the entire system's behavior. (Systems dynamic society 2012, Radzicki & Taylor 1997).

Different kinds of systems can be studied with SD, for example, models analysing population growth, effects of certain policies in different areas, market changes, spread of diseases in a population, and economic systems both in private and public sectors (Radzicki & Taylor 1997). SD models and other types of simulation models are often used to help in problem solving and decision making (Sargent 2008, p 147). For modelling the spread of different diseases, a certain kind of epidemiological SD model is often used: the SIR-model (Susceptible - Infected - Recovered). (Towers et al 2012, p 415)

The A(H1N1) influenza pandemic of 2009, has been studied before by means of SD. In 2010, researchers at Department of Computer and System Sciences at Stockholm University (DSV) created a SD model called VirSim[1] (Fasth, Ihlar & Brouwers 2010) in order to investigate the spread of the disease in Sweden during the influenza season of 2009-2010. Other SD models were made in other parts of the world as well, with varying scope and level of detail (Pruyt & Hamarat 2010). VirSim, however, focuses on showing how different strategies dealing with the pandemic would affect the outcome of the 2009/10 influenza season; it does not study the effects of the mass vaccinations on the following season.

Other studies regarding pandemic influenza spreading using SIR-models have been done.
Towers et al (2012) studied the impact of school closure on pandemic influenza by using a seasonal SIR-model. A seasonal SIR-model means that the model takes into account the seasonal variation of transmissibility of influenza, by using a modifying function that changes certain parameter values over time (Towers et al 2012, p 413).

Furthermore, a microsimulation model called MicroSim was developed in 2004 at SMI

(Brouwers, Mäkilä & Camitz 2006, Saretok & Brouwers 2007). The goal was to produce a tool for testing the effects of different intervention policies. MicroSim represents nine million individuals living in Sweden, and was first developed to simulate outbreaks of smallpox, but was further developed in 2006 to support pandemic influenza simulation. Microsimulation models differ from SD models in several ways. While SD models display systems at a macro level, microsimulation models display them on a micro level. They have a low level of abstraction, and can model population on an individual level, taking in account advanced contact patterns, such as individuals going to work etcetera. MicroSim studied what the effect on the 09/10 influenza season would have been if the mass vaccinations did not take place. However, as in VirSim, this model focused on one season only, and did not investigate the effects on the following season.

## 1.2 Problem statement

The mass vaccination against A(H1N1), commonly referred to as "Swine Flu", in Sweden during the fall/winter of 2009 has been issue of much debate. This debate was fuelled after the pandemic turned out not to be as serious as WHO and the Swedish government had feared (Engwall 2009, Ledarsidan DN 2010). Additionally, the vaccine was later found to have caused narcolepsy among other side effects (LMV 2012).

All consequences of the vaccinations have not been completely investigated yet, but, for the sake of future policy making for pandemics in Sweden, it needs to be done. A project group at Swedish Institute for Communicable Disease Control (SMI) are currently doing research to evaluate the vaccination program and study its effects. One aspect of the vaccinations, which needs to be investigated by this project group, is the protection the vaccination program provided to the population during the following influenza season of 2010/2011. The degree to which infections during the subsequent season could be avoided is, from here on, referred to as "bonus effects". (SMI 2011:a)

The effects that mass vaccinations might have on the following seasons infections, represents a gap in epidemiological knowledge and research. When this gap of knowledge is filled, it will provide valuable insights that can aid in policy and decision making, thus have consequences in several different fields, such as public health care, economics and pandemic preparations. (SMI 2012:a)

One vital problem is how to be able to measure these bonus effects. Since these kinds of questions and issues are not possible to test in real life, the use of simulation models can be considered suitable. However, simulation models, such as SD-models, have not been used for this specific purpose before. It is unknown if, and how, suitable they are for this purpose. One reason for this is that SIR-models tend to only study a single epidemic outbreak, but in order to be able to study the bonus effects of mass vaccination, two outbreaks must be implemented in a single model (SMI 2012:a).

This thesis will investigate if SD-models are suitable for providing information about bonus effects. Thus, it may aid in filling in the gap in epidemiological research, as well as the gap regarding the use of simulation models.

**Research question**

**Purpose (of thesis)**

**Goal**

**Methods: Qualitative/ Quantitative, Inductive/Deductive, Literature study**

# 8 Conclusion

To be able to answer the research question, the model needs to be able to provide three valid output values for two different scenarios; Infected 09/10 with mass vaccination, Infected10/11 with mass vaccinations and Infected 09/10 without mass vaccination. Valid output values means that the output matched the historical data and the output value from MicroSim. In addition, it was required that the model passed all other validation tests in order for it to be considered valid.

When these three output values were deemed valid, it became possible to measure the effects on 10/11 without mass vaccination, and thus draw conclusions about bonus effects following influenza mass vaccination.

Since the SIR-model discussed in this thesis has strong support for it being valid, and it is able to supply the required information, the conclusion that can be drawn is that SIR-models can indeed be used to provide information about bonus effects following influenza mass vaccinations. ...not as individuals, there were some problems in tracking which individuals got vaccinated and then sick again the following season. The situation of not being able to see the model at an individual level DQG ³NHHS WUDFN´ RI LQGLYLGXDO entities can be seen as one of the limitations of SD models. Since this sort of real world-system is very sensitive, it might be necessary to construct a more granular model. It might be the case that the output of a SIR-model is also not as detailed as the output of other types of simulation models, such as microsimulation models. Geographical and social aspects are possible to implement in an agent-based model, which probably would provide even more information about the spread of pandemic influenza.

## Evaluation of the Research

Other issues:

“To be able to answer the research question, “

Is the purpose fulfilled?

GOAL? Is the goal accomplished?

What is the result of the degree project? are the results? Of the thesis?

Is the own work related to literature study and related work (chapter 2)

# CASE 2: Security Evaluation of the Electronic Control Unit Software Update Process

## 1 Introduction

Modern heavy-duty vehicles rely greatly on software running the embedded devices controlling their different modules - Electronic Control Units (ECUs). This thesis analyses the process of updating software on ECUs for security flaws, and is conducted in the global heavy-vehicle company Scania. This chapter introduces the background, purpose and general outline of the thesis project.

### 1.1 Background

Scania Group [40] is a global company, which produces a wide variety of trucks, buses and engines. They provide a complete technical, educational and financial service of managing an inventory of motor carriers. Among Scania's services, workshop maintenance is the one relevant to this thesis. It is during a mainte- nance checkup that the latest software is installed on the vehicle's computational units. Scania is responsible for the correct functioning of the vehicle, which ad- ditionally translates to the safety of its driver and passengers. Therefore, it is in Scania's interest to ensure the integrity of the installed software.

A modern vehicle holds in itself an array of separate modules: the engine, breaks, air conditioner, locking, and dozens of others. Each module in a vehicle consists of mechanical components, as well as electrical sensors and actuators. An Electronic Control Unit (ECU) is an embedded system, which manages and controls a module's electrical components by running software stored in its non-volatile memory [20] [22]. The ECUs in a vehicle communicate over a Controller Area Network (CAN), a message-based communication protocol [9]. This network can be accessed externally via a standard connector, which is commonly located under the dashboard [34]. During routine checkup, the CAN is connected to for diagnostic purposes: reading usage and performance data, and updating the software of ECUs [43] [40]. To make these processes
easier, specific application-level communication standards are in use: Keyword Protocol 2000 (KWP2000) [44] and Unified Diagnostic Services (UDS) [42]. The hardware for connecting to a vehicle's internal electrical system is available and respective diagnostic communication standards are open as well. Therefore, anyone can set up an interaction with a vehicle that they have physical access to. KWP2000 and UDS define security measures, in order to guarantee that only authorised persons get access to read and write ECU software data.

### 1.2 Problem Description

ECUs are responsible for most of the functionality of a vehicle. Software bugs and miscalculated settings in an ECU may easily cause a vehicle to malfunction. This does not only pose an inconvenience for the business it serves, but could also potentially endanger the safety of its transportees or other travellers on the road. Therefore, it is of great importance that before a vehicle is handed over to the customer, the functioning and co-operation of all its modules is thoroughly tested.

Some parties may have interest in altering the software of an ECU - for example, a truck owner wishing to improve some performance indicators, or a criminal intending financial or physical harm. Unexpected behaviour of a vehicle resulting from untested software alterations may then lead to legal warranty cases. In such situations it may be difficult to prove that any manipulation has occurred, and the responsibility would lie on the manufacturer. Therefore, it is in the interest of the manufacturing company to minimize the possibility of ECU software being altered outside its own production units and verified workshops.

The industry standards concerning ECU software updating feature several security measures, but the degree of their implementation varies from device to device. Previous research has already demonstrated cases of overriding ECU software security in automobiles. However, lack of overview exists when it comes to the security of ECUs used in trucks and buses.

**Research question**

**Purpose (of thesis)**

**Goal**

**Methods: Qualitative/ Quantitative, Inductive/Deductive, Literature study**

# 8 Conclusions

Modern vehicles are controlled by a distributed computer system of embedded devices - Electronic Control Units (ECUs). Customising the functionality of a vehicle comes down to changing the software on ECUs. To avoid problems with road safety, as well as legal issues, it is desirable that only the vehicle manufac- turer is able to make software alterations to ECUs controlling critical modules. This goal requires for appropriate security measures to be in place, so that ex- ternal parties would be unable to get access to an ECU's self-reprogramming functionality. Previous research has already identified several potentially ex- ploitable vulnerabilities in the diagnostic interfaces of ECUs used in automo- biles. Also, general-purpose attacks have been demonstrated, mostly related to sending immediately executable commands to ECUs.

This thesis continued the research by moving the focus to ECUs used in heavy-duty vehicles, and specifically the reprogramming functionality of the ECUs' diagnostic interfaces. The purpose was to evaluate the process of per- forming software updates on different Scania ECUs from the security perspec- tive. As a result, it reports security vulnerabilities, which may lead to an unau- thorised person flashing ECUs with arbitrary software. To thoroughly present vulnerabilities and their impact, the thesis has three consecutive goals: to iden- tify vulnerabilities, demonstrate attacks, and propose solutions. The research was carried out as a quantitative study, using experimental research methods and a deductive approach. Experimental methodology was used to test ECUs and calculations were performed to arrive at conclusions. To evaluate the sever- ity of the vulnerabilities found, the results were validated by performing exper- imental attacks.

To start off the security evaluation and meet the first goal - identifying vul- nerabilities - formal software testing methods were used. The abstraction level tested was the integration of the ECU and a diagnostic node. The subject of testing was the application-level diagnostic interface of each chosen ECU. The vulnerabilities pointed out in previous research, as well as the security require- ments defined in the UDS and KWP2000 standards documents, were built upon to compile a set of test cases. Each test aimed to verify the correct or sufficient implementation of a functional security requirement. Positive tests were exe- cuted to verify the presence of standardised security requirements, and negative testing was added to identify any security bugs. As the specification documents and software code used internally in Scania was available for this study, then grey-box testing methods were applied to save time. To make accurate mea- surements of time delays and query speeds, some tests were automated with scripts.

## 8.1 Identified Vulnerabilities

The results of the security evaluation reveal several exploitable vulnerabilities in ECUs with varying security strength. The main identified problems were the following:

- No security implementation whatsoever. These systems were not chosen to perform further attacks on.
- Short seed and authentication code used in authentication. It is feasible to perform a brute-force attack on an ECU with a seed and authentication code length of merely 2 bytes, and an average-length failed access attempt delay.
- No authentication or encryption of messages carrying flashing data. Since CAN is a message-based protocol, then lack of higher-level authentication and sending plaintext messages makes it vulnerable to man-in-the-middle attacks.

The found vulnerabilities are in line with previous research, which empha- sizes similar or related problems in ECUs used in automobiles. To determine and demonstrate the significance of these vulnerabilities, they were further ex- perimented with.

## 8.2 Performed Attacks

Experimental attacks were devised to meet the second goal of the thesis. The aim of this was to validate whether the identified vulnerabilities could realis- tically be exploited to flash an ECU with arbitrary software. The goal of the attacks was to gain unauthorised access to an ECU's flashing functionality.

Three different types of attacks were successfully conducted, implying that the identified vulnerabilities pose a realistic threat.

### 8.2.1 Brute-Force Attack

Previous research suggests that short seeds and keys might only provide a tem- porary protection from malicious security access attempts. As in the course of testing an ECU was found, which featured a 2-byte seed and authentication code, the realistic feasibility of a brute-force attack could be verified.

A random 2-byte combination was chosen and repeatedly used as the authentication code. New seeds were queried, until the authentication code was accepted, and a higher security level was unlocked. This experiment was con- ducted twice, and succeeded in a feasible amount of time on both occasions.

Gaining access to a higher security level makes it possible to successfully send flashing commands to an ECU. However, flashing itself was not performed as a part of this experiment - bypassing authentication and unlocking the ECU was the sole goal.

### 8.2.2 Man-in-the-Middle Attack

Previous research also points out that the message-based nature of the CAN network, and the lack of encryption in ECU diagnostic interface implementations makes it susceptible to message replaying. It was the intention of this research to verify whether it could be exploited to perform a man-in-the-middle attack.

This experimental attack was performed on an ECU, which in the course of testing proved to be one of the most secure ones. However, flashing messages are sent without separate authentication of each message, and software data is unencrypted. This means that another device can be placed between the communication line of the tester and the ECU, to relay messages between them, and change or drop them at will.

The testbed consisted of 2 computers - one of them acting as the tester, and one of them the adversary - and an ECU. The adversary managed to successfully relay messages to the ECU until authorisation was successfully finished. Then, it dropped the following messages from the tester, and replied to them with positive response messages and calculated checksums. At the same time, it sent messages containing arbitrary software data to the ECU. Once the tester and the adversary had finished sending flashing data, the adversary continued relaying messages from the tester to the ECU, in order to finalise the flashing process, and not arise immediate suspicion. As a result, the legitimate workshop flashing tool finished its task without giving error messages, and the ECU was actually flashed with arbitrary software provided by the adversary.

### 8.2.3 Combined Attack

The previous attack is defined as an active man-in-the-middle attack, since the adversary intercepts messages. However, the intention of the adversary may simply be to sniff on the communication between the tester and the ECU. This way they could find out a legitimate seed-authentication code pair, and use it in a sped-up brute-force attack to unlock a higher security level. The combined attack is suggested as a part of this research.

The same test set-up was used as in the previous attack. The ECU used was the same as in the first brute-force attack. This time, message relaying was not stopped. Instead, when the tester and ECU performed a SecurityAccess challenge, the seed and authentication code were sniffed and stored. Later, the ECU was queried for seeds without sending any response, until the sniffed seed appeared. Then it was replied to with the sniffed authentication code. Since no false authentication codes were sent, then time delay was never activated. The higher security level was successfully unlocked with substantially reduced time in comparison to the previous brute-force attack.

## 8.3 Proposed Solutions

As the identified vulnerabilities proved to be exploitable, solutions were also proposed to avoid such attacks on ECUs in the future.

First, it is essential that ECU software implements the security functionality suggested in the UDS and KWP2000 standards. Flashing functions should only be available after unlocking a higher level of security via a successful completion of the SecurityAccesschallenge. The seed and key used in the challenge should be 8 bytes long to mitigate brute-force attacks. In case of a shorter key, a time delay should be implemented after a failed security access attempt, before a new seed can be queried, as well as after every boot. To prolong the time required to performed the combined attack, a time delay should be activated right after a SecurityAccessseed is queried.

Secondly, the ECU software update process would benefit from using the SecuredDataTransmission service for sending software data to the ECU in an encrypted form. This would protect the proprietary software of the vehicle manufacturer from being reverse engineered. Also, it would be impossible to reuse the software data to illegitimately flash another ECU. Additionally, the software data messages could not be altered by a "man in the middle".

Since decryption on a large scale might be too labour-intensive for an ECU, an alternative protection method against an active man-in-the-middle attack would be adding a signature on each message containing software data, con-

firming that it came from a legitimate source. The signature would consist of a message hash and a nonce, encrypted with a session key. The session key would be agreed upon with the Diffie-Hellman algorithm, where the tester is authen- ticated with public-key cryptography. Alternatively, if symmetric cryptography is preferred, the session key could be a random number chosen by the ECU, encrypted the shared secret key used for SecurityAccess.

## 8.4 Constraints

The research only focused on one part of the ECU software update process - the security implementations in the ECU's self-reprogramming interface. In that component of the whole process several vulnerabilities were found and demonstrated. It may very well be that additional security holes lie in the other elements of the process, such as the storage and distribution of cryptographic keys or production and signing of flash files. Additionally, no offline brute- force attacks were performed on the cryptographic algorithms to uncover secret authentication codes, although they may have been feasible.

Therefore, this research does not present a complete picture, regarding the security of the ECU software update process.

The experiment performed in the course of this thesis relied greatly on the internal specification documents and proprietary software code of Scania. The documentation was used to build an understanding on the Scania-specific details of the application-level communication protocols used in updating the software of different ECUs. These details could theoretically be reverse engineered by an adversary, who has sniffed on a flashing procedure, therefore this should not completely nullify the reproducibility of the research. Scania's proprietary software code was used in experiments to take care of establishing a connection with the ECU, as well as for performing the flashing itself, after authentication was bypassed in the course of the man-in-the-middle attack. As Scania software is built on the publicly available Kvaser CANlib library [29], then, given suffi- cient time and knowledge about the flashing protocol, these scripts could also be reproduced without access to proprietary software.

The brute-force attack and combined attack did not follow up with software updating after authentication was bypassed. Thereby, the attacks are slightly incomplete, although the potential threat has been demonstrated.

All tests and experimental attacks were performed in a testbed, not on a real vehicle. ECUs were treated as standalone devices, not as a part of a complete system. Adding a "man in the middle" to a truck may mean inconspicuously inserting a pre-programmed device in its internal electrical system. Also, addi- tional restrictions may apply to perform these attacks or make arbitrary software functional in a complete vehicle.

The results of Test case 4.1 were only concerned with randomness as per- ceived by a human observer - that seeds did not have an obvious pattern. Given the context, this may have been a sufficient criterion, but it would have been more comprehensive to analyse the pseudo-random number generation algo- rithm used, to see whether its results can be predicted.

## Evaluation of the Research

# CASE 3: Mobile Multiplatform Web Application Development for Online Product Services

# 1. Introduction

Today, a large portion of mobile phone users own a Smartphone [1, 2]. Having facebook, twitter, angry birds and other mobile applications available all the time are taken for granted by many. Of course this has not always been the case.
A few years ago software development for mobile devices was in its cradle. With the launch of Smartphones the mobile software market has been rapidly expanded. As in any healthy market there are several competing companies with their own Smartphones and operating systems. The multitude of available mobile platforms has allowed third-party mobile software companies like Rovio, the creators of the popular game "Angry birds" (http://www.rovio.com), to flourish. All of these third party companies have one thing in common: the challenge of multiplatform development [3].

## 1.1 Background

Every company wants to maximize their profits. One way to do this is to maximize the number of platforms the company's product can be launched on. This means that the product, application in this case, needs to be compatible with multiple platforms. Mobile multiplatform compatibility comes in three flavors: native, web, and hybrid [4]. The native approach uses coding languages and frameworks exclusive to a particular platform and in essence means that an application is rewritten from scratch for each platform the company wants to support. The web approach is an attempt at Java's "write once run anywhere" philosophy: only one application will be developed which can execute on any device and platform [5]. This is possible since all modern mobile platforms support the same web markup-, script- and preprocessor-, languages, essentially running the application in their respective web browsers. Finally the hybrid solution is a combination of the native and web approaches. A hybrid application offers many of the functionalities of a native application while at the same time keeping the bulk of the code in web languages. A hybrid application is thus a shell of native code encapsulating a pure web application.

This thesis is about the development of a mobile multiplatform application. The mobile application will be an online retail service and supports an undefined number of mobile platforms. This leads to the critical decision of a compatible multiplatform development strategy.

## 1.2 Problem

Selecting a target platform is arguably the most important decision to a mobile software development project since it will not only govern the target audience but also the development environment and the required expertise to complete the project.

**Research question**

**Purpose (of thesis)**

**Goal**

**Methods: Qualitative/ Quantitative, Inductive/Deductive, Literature study**

# 8 Conclusions and future work

The thesis has presented an example of multiplatform app development by following agile software development methods. The thesis has described the decision making process for selecting an appropriate multiplatform development strategy for use in a development project.

- Create a multiplatform mobile application.
- An app was developed and the development process is described in the thesis.

The web strategy is suitable for companies like SPS that have an existing website to be made into an app. The decision making process is likely to be quite different for companies with other services and goals from those described for the example company SPS. The SPS Company was modeled after a medley of real services but not one in particular. Therefore, the design and implementation decisions are applicable for real companies. Even though the web app created is not a final product ready to be released it is still a working application that can be further developed into a final product. The app is fully operational and designed with mobile devices in mind. The app allows a customer to use the app just as they use the SPS website. The description of the development process is based on the authors` experience. Attempts to replicate the work may lead to a different development process. One of the most important parts of software development experienced is the value of having a good planning phase before the project starts, as well as good planning for each iteration during the project.

The biggest problem encountered due to the limited pre-planning of the project was the fact the database had to be altered at several times during the development. This resulted in rewriting of code which could have been avoided if the underlying database (tables, field names, value types) had been determined in an early stage of the development.

Disciplined use of the version control system allowed the developers to focus on actual development rather than management of the source code. Since the version control handles backup and merging of source files.

When developing web apps with JQuery Mobile it was discovered, belatedly, that full page reloads have a negative impact on performance. This is something that the developers learned to avoid as much as possible, however, the app was already heavily dependent on PHP scripts with full page reload requirements. The full page reload problem can be solved by restructuring the app navigation and inter-pages message passing. On mobile devices with high internet connection speeds and good processing power this issue is a minor one. However, it is very noticeable on older mobile devices or where there is limited internet connection available.

## Evaluation of the Research