Figure 1: Computer as judge, possible or not?

CAN COMPUTERS DECIDE WHAT IS LEGAL AND ILLEGAL?

Jacob Palme
Department of computer and systems science
Stockholm University
Skeppargatan 73
115 30 Stockholm
Sweden

ABSTRACT

Humans are able to understand that rules must not be adhered to 100 % all the time. There are special cases, where the rules should not be adhered to. Computers do not have this ability. A society where computers are judges, will not be a nice place to live in. This papers illustrates with practical examples why computers should not be judges.

KEYWORDS

Computers, Control, Rules, Laws, Legislation, Judges, Courtroom

1. INTRODUCTION

The main theme of this paper is that one should be very careful with programming computers into becoming judges. The reader may react with the question: OK, we should not program computers into becoming judges. But why is this such a big issue? We could program computers to do lots of useful things, even if we don't make them into judges. Why write a paper on why we should not make computers into judges, why is this such a big issue?

Published in Information and Communicaton Technologies, Society and Human Beings, IGI Global, Hershey PA. USA, November 2010.

Well, if you think a little more, you will find out that it is extremely common that computers are some kind of judges. Not in the common case of judges who work in courts and make verdicts. But it is very common that rules about how humans should behave are written into computer programs, and that the programs are written in such a way that their users are forced to adhere to these rules. The people who design and write computer programs will in many cases be rule-makers. By the way they design the programs, they decide which rules are valid among people who use these programs. Computer programs are full of detailed rules about how the world should work.

- Messages may have a limited length.
- It may be illegal to include pictures in certain messages.
- In order to use the software you have bought, you are forced to sign a checkbox saying that you agree to long, detailed "conditions of use", which you have no option to negotiate in any way, and which probably are very unreasonable. If you do not check that box, the software you have bought will not work.
- To perform a certain action in the administrative system at your workplace, you have to fill in long detailed forms with information, which makes the systems cumbersome to use, and which are not really necessary.
- Requests for permission do not allow you to specify the reasons for your request in the way you would prefer.
- Etc, etc.

Life is full of these minor or major obstacles posed by computer systems. They are, in fact, rules. They specify what you may do and not do. Often in ways which are cumbersome and unnecessary complex. Because the rule-makers, the designers of the software, want to impose their rules on you. Real life is full of such rules. But when the rules are written on paper, you have the option of doing things in simpler ways. No one will usually check or require you to adhere to all rules, fully and in every detail.

A well-known method for work-force conflict is that everyone does everything exactly according to rules, making things much more time-consuming so that tasks are not ready in time. In reality, we ignore many rules when we feel they are too complex or unnecessary, if we follow all rules in every detail, work grinds to a halt.

But when the rules are enforced by computer programs, there is no option of making life simpler by not always follow every directive exactly. And these are examples where computers act as judges, forcing you to exactly abide by every rule. And that is why computers often makes life unnecessarily complex.

2. EXAMPLES OF RULES IN REAL LIFE

Example: A breath analyzer (Swedish: Alkolås) is a tool to prevent drunken driving. The driver breathes in the analyzer. If there is too much alcohol in the breath, the driver is not permitted to start the car. But there can be special cases. Suppose the husband has a heart attack or a stroke. His future survival depends on getting him to the hospital rapidly. His wife has drunken a little. In this case, it might be wise to let her drive him to the hospital in spite of having a little alcohol in her breath.

In fact, one way of installing a breath analyzer is not to let it stop her from driving (*stopper*). Instead, it will report the violation, and a real judge, afterwards, checks if her driving should be permitted in this special case (*reporter*). This may be a better way of implementing the breath analyzer, because the human can understand the need to diverge from the rule in this special case.

Letting the breathalyzer stop her from driving is a case of making a computer into a judge. And letting a human afterwards decide if she was right in driving in this exceptional case may be a better way of implementing such a rule. In fact, breath analyzers have been implemented in both ways, both as *stoppers* and as *reporters* of violation.

Some people may claim that the computer was not a judge. The human who decided that the breath analyzer should work as a *stopper* was the real judge. But in reality the computer, if implemented as a *stopper*, will act as a judge and may cause the husband to die in the special case where his life depends on getting him to the hospital rapidly. There may be cases where it is best to let a computer stop illegal behaviour. But one should consider carefully if a *reporter*-type of implementation may be better than a *stopper*-type of implementation in cases like this.

3. THE REAL USE OF LAWS AND REGULATIONS

Why do we have laws and rules? One way of understanding this is to say that laws and rules are ways communicating experience and practice. People with a lot of experience with issues will write the rules in order to help people with less experience to act in the best way. Looking at laws and rules in this way makes it easier to understand why rules interpreted by humans are more acceptable than rules interpreted by computers. Sometimes, it is necessary to set up courts and other systems for enforcing the rules. This is not always necessary. And the very hard way of programming adherence into computers may be even less necessary.

Figure 2: Child runs in front of car

An example of a rule in real life is the rule saying that you may not pass a street-crossing against a red light. This may be a good rule to follow in real life. But what if a child is starting to run across the street, in front of a car. Then you may save the life of the child by stepping out and stopping the child. And no one will stop you or prosecute you for this violation. Life is full of such cases where it is sometimes better to not follow every rule exactly. But if the rules are enforced by technical systems, computers, you do not have this option, you have to follow every rule exactly, all the time.

People who create computers, and who build rules into the systems, do not always understand the dangers of always enforcing every rule, all the time, using the computers to enforce every detail of every rule.

4. DETAILED FORMS

Everyone has encountered situations where you have to fill in a form to get something done. And where you do not have to fill in every detailed field every time. Someone says: “Just sign your name here”, and you do, and things gets done, even if you have not filled in every field in the form. But if the filling in of this form is controlled by a computer, you may always have to fill in every detailed field. The every-day rationalization which we do so often is not possible. Sometime, of course, it is important to follow the rules, like point the hose at the base of the fire before pressing the lever to release the fire-extinguishing foam. But sometimes the rules contain details you can skip to get things done faster.

5. EVERY-DAY RATIONALIZATION

In many workplaces, people will perform tasks more efficient each year than the day before. This is called “every-day rationalisation”, the optimization of tasks done by doing things in simpler and more efficient ways. But if tasks are done by the use of computer, the computer will sometimes prevent such every-day improvement in doing things. [Hoare 1975, Palme 1997].

Sometimes, it is important that people do perform every detail and not pass over any step. Air-line pilots have checklists, in order to help them remember to do every check on the list. This is important for safety, to prevent accidents. In such cases, technical means of ensuring that people do perform every step are sometimes used, just because such technical means ensure that you do not skip any step. Night watchmen have special keys they have to put in locks when they walk around the building. These keys ensure that they do not forget, through laziness, from walking around the building. But if the night watchman hears a sound which should not be there, he will of course hurry to the place of the noise, to check if someone is trying to break-in. The technical systems does not prevent him from diverging from the normal route, even if this means that he will not click the keys in the right order that night!

In other cases, it is not really necessary to do every step. And the human ability to rationalize everyday life by skipping certain actions is useful and makes life simpler. I have a baking recipe for baking biscuits, which specifies that you have to add ingredients in a certain order, then mix, then add more ingredients. I have found that the biscuits are just as good when I just throw all the ingredients together and start the baking machine. This is a kind of everyday rationalizing which I have made for the baking of biscuits. And in this case, the machine did not prevent this rationalization.

Figure 4 shows how people sometimes circumvent technical restrictions which they have found not necessary. I am sure you have seen many other examples from your own life.

Sometimes this may be necessary. For example, there is a human tendency to stop performing actions which are necessary only to avoid seldom occurring risks. Example: A pilot forgets an item on the pre-flight check list, or a night watchman forgets to go to a normally empty part of the building. In such cases, it may be necessary to use technical means to ensure that the human follows the rules, for example the night watchman must turn a key to show that he has passed that part of the building. But this does not forbid the night watchman from disobeying the rules in special cases, for example skip the empty corridor if there is a thief in another part of the building. The danger is when the computer does not allow you to do things in other ways than those foreseen when programming it.

People are in fact very clever in circumventing restrictions in order to do what has to be done. See for example Figure 4. I have searched on the internet, and found hundreds of copies of this picture on different web pages.[1] Obviously, many people have felt that this picture shows something important!

[1] In order to find the original creator of the picture and pay for the usage of the picture in this paper. I have not been successful in this.

Figure 3: Human ingenuity in circumventing mechanical obstacles.

6. BALANCED COMMUNICATION IS BETTER THAN ONE-SIDED

Power is addictive [Hoare 1975]. By this is meant that people who are put in a position of power, will tend to utilize that position to favour their belief in what is the proper way of doing things. There is nothing evil in this, it is just that they believe they have good competence in how things should be done. If they are given the task of designing computer software, they will want to use their competence to design the computer software according to their knowledge about how things should be done. This is a kind of communication process. They are communicating their competence in how things should be done in the design of the computer software just as rule-makers communicte their knowledge of how things should be done in the rules they make. The danger with this is that it is a uni-sided communication process. It is only from the designers, to the users, not bi-directional. It is a communication process promoting a feeling of helplessness, which is known to cause depression and dissatisfaction [Seligman 1975]. And uni-directional communication processes are almost always bad. Good communication processes are almost always bi-directional. Only bi-directional communication gives the richness of give-and-take, argument-and-counterargument, my idea-against your idea, which are characteristics of good communication processes. Bi-directional communication processes mostly are characterized by I say you have no say, Listen but not speak which are characteristic of undemocratic, dictatorial mode of communication. Not only will software design per se be one-sided unidirectional communication processes, but the communication processes provided by computer software designed by people in power, tend to be unit-directional.

Now you may say as counter-argument, that “My company has a very democratic process of designing software with committers representing all user groups overseeing and guiding the software design”. Or “My company have design groups consisting of representatives of our customers who oversee software development”. And if a company have such democratic committees, surely, things are not bad! Wrong! As son as a person is appointed into such a group for overseeing software design, that person becomes a person of power (assuming that you really give influence on software design to these groups, and that these groups are not token groups with no real power) will immediately become a person of power, a person for who all the dangerous thought processes of people in power will crop up as described in the beginning of the previous paragraph. Win-win solution which are characterized of bi-directional communication processes are difficult or impossible with uni-directional processes [Harris 1969, Gordon 1970].

My cable-TV company has a web form for changing my subscriptions. All changes which give them more income are easy and simple to do using the web form. All changes which give them less income (such as unsubscribing from a channel) are impossible in the web form, and force me to write a postal letter to them with my request for a change. If, for example, I want for one month add a channel to my subscriptions, I will have to perform a add channel operation (easy to do using their web form) but if I want to reduce my cost I have to perform a subtract channel operation (can only be done by postal mail) and which is one month delayed. It is of course not by mistake they have designed their system in this way. They hope that I will be lazy and perform only one of these two operations, causing them an income of many months of subscriptions to this channel, when I i reality only wnated to see one particular program.. This way of designing the web dialogue is useful in companies which want to maximize their profit by making all user actions which increas in the costs to the user easy to perform, and all decreases of the cost to the customer difficult and cumbersome to perform.

It is not by chance that this problem occurs with a cable-TV company. Cable-TV compaies are monpolies with lots of power. Such companies have very high capabilities to control free speech, since it is so diffiucult or impossible to switch to a different cable-TV company. They utulize their position in power to increase their revenues.

7. USER INFLUENCE

Human beings hade a need to be able to influence their life. They will be more happy and satisfied, and will be able to do a better job, if they can influence their life, and use their abilities to perform their tasks better and better.

Conventional solution: Let the users influence the development of the software system they are going to use.

Corrective action: Users require new features of the software, developers are overloaded with work to adjust the software, there is a huge backlog of tasks, the software gets more and more complex through many haphazard extensions.

Alternative solution: Design the software so that the users can, themselves, modify it according to their present and future needs.

Corrective action: Educate special so-called "local experts", who work locally in the local user groups, and help users with extension of the software to their needs.

8. A TOO DILIGENT PORN STOPPER

A funny example of how misuse of power can occur in a communication system.

A person I know sent a message containing a part of a script to a friend through a messaging software provided by people “who wanted to do good”. The script in his message contained the following too lines:

```
#define one 1 /* foo menu */
#define two 2 /* bar baz */
```

And it arrived corrupted in the following way:

```
#define one 1 /* foo me */
# fine two 2 /* bar baz */
```

What had happened was that the “be good”-software in the messaging system censored all messages containing porn, and the word nude was regarded as indicative of porn. If you look carefully, you can see that

two last two letters in the first line and the first two letters of the second line makes the word “nude”. Porn, cries the friendly software and corrupts this script, which had nothing at all to do with porn.

In general, designing programs which pass through only “good” messages and no “bad” messages (according to your particular idea of what is “good” and “bad”), are very difficult to design. Dictatorial countries like “The People's republic of China” and “Singapore” make attempts. Programs to prevent the word “sex” have difficulty with “Middlesex”, a local government in England. And should you stop information about breast cancer because it contains the word “breast”?

9. IS THE INTERNET ILLEGAL

Usage of the Internet is in fact illegal according to the privacy projection laws in many countries. These laws makes it illegal to send personal information unless in special restricted ways. If I write a letter to my brother saying that our mother is going to try out a new miraculous cancer treatment method, with large success probability, this happy letter was probably illegal according to Swedish law for many years. It contains personal information about an easily identifiable person. And the personal information is medical information, which should only be permitted, according to the privacy protection laws. I ran one of the first BBS-es in Sweden starting in 1978. My BBS was forbidden by the Swedish Data Inspection Board, saying that it was against the Swedish data inspection law. A few years later, an author was forbidden by the Swedish Data Inspection Board, from writing a book. In 1978, I was forbidden from running one of the world's first e-mail systems, because it allowed people to send personal information to each other. Nowadays, through strong acions of lobbying, I and people who understand the dangers with laws on computer usage, have succeeded in changing the Swedish laws, allowing such applicatons like e-mail which was previously forbidden.

Actually, almost all usage of the Internet is illegal according to the privacy protection laws in many countries. In later years, the Data Inspection Boards have become a little more cautious. But it is very difficult to design laws making the internet illegal “by mistake”. that is because the privacy protection laws are basically against certain kinds of “free speech”, and internet is by design a system for “free speech”.

10. MUST THE COMPUTER STOP ALL UNWANTED BEHAVIOUR

There is a tendency, among designers of computer systems, to believe that the ideal computer system is a system designed so that it guides people into the correct and good behaviour and prevents all unwanted behaviour. The danger with such systems is that they are too controlling. In real life, one often has to step through steps which are incorrect on the way towards a correct solution. Look at the mostly used of all computer applications. These are:

- Word processing software for writing texts, like for example Microsoft Word.
- Spread-sheet software such as Microsoft Excel.
- Web browsers for looking at the world wide web.
- E-mail software for sending messages.

All these software systems are based on giving the user maximum freedom. Microsoft Word is not successful by preventing “incorrect” texts, but in giving freedom to write whatever the user wants. Spread-sheet software are successful by giving maximum freedom in designing different kinds of calculations between cells. E-mail is not successful by forbidding the writing of “incorrect” messages but by permitting any kind of message with any kind of content. Web browsers are successful by the freedom of looking at all information all over the web. Successful designers of software systems are not successful by designing software which prevents illegal usage, but by designing software which gives maximum freedom of usage.

When discussing software design with people doing the design, it sometimes seems as if they are of the opinion that good software must stop all kinds of incorrect usage of the software. The designer's knowledge of the task must be used to prevent all incorrect usage.

A reverse variant of this belief is the belief among some computer crackers that everything which was possible to do must be permitted. “It was your fault”, says the cracker, “by making it possible, you made it

legal". In fact, lots of things which can be done using a computer are still illegal. Making the software to allow something does *not* make that action permitted or legal.

I have noted this often in discussing regarding the design of the administrative system we use at our university do manage courses, lecture room allocation, student marks, etc. For example, it is not correct for students beginning discussing their thesis writing before they have achieved success in most other courses. The system enforces this, and refuses to provide administrative support for students who want to start their thesis work earlier. However, it is often useful to base a thesis on tasks in earlier courses. This is hampered by the system. This is just one of many examples of how the systems causes unnecessary trouble by enforcing unnecessary rules regarding how to write a thesis. Let the tutor, not the computer, decide when a certain student can beging working on his thesis!

It is not necessary to put all kinds of control of incorrect usage into computer systems. Instead, it is best to let people decide what is correct in their usage. Humans are better at checking what is right and wrong than computer software.

11. CALENDAR SCHEDULING

One example of a kind of software system which easily becomes unnecessarily restrictive is calendar systems, i.e. software for managing the allocation of times for meetings, etc. Such software can incorporate lots of rules regarding how *good* calendaring should work. It might be better not to put all kinds of rules into such software. And if such systems have to manage certain kinds of issues, it can perhaps better be managed by letting the software *warn* of possible problems rather than forbid them. Human can then decide whether to take action after such warnings, instead of letting the software take action in usually very clumsy ways. For example, certain meetings may be more important than other meetings. It may be necessary to reschedule other, already scheduled meetings, to assign time for other, more important meetings. But it may not be ideal to design the software to handle this. It might be better to have *humans* decide whether to take action because of such rules or not. All rules must not be built into the software. Some rules may not be suitable to put into the software. For example knowledge that meetings with Mr. Jones should never be scheduled on Monday mornings, may not be nice to put into the software. Calendaring systems are typical of systems where you can put any number of special rules into the software, but where this may not always be suitable or even possible.

12. CONCLUSIONS

- The successes of human society is based on the flexibility of humans and their willingness to adapt their activities to different circumstances.
- Humans are most happy and productive if they can influence their living environment and contribute to solving problems together.
- Laws and regulation are a form of communication between humans. They are in reality only guidelines, people have to adapt to varying circumstances and interpret and apply the rules with understanding and human compassion. If everyone had to adhere 100 % to all laws and regulations, human societies would not work any more.
- This is usually no problem when the laws and regulations are written on paper. But if the laws and regulations are programmed into computers, so that the computers control what is allowed and not allowed, serious problems will often occur. In the best case, people will only be unhappy and unproductive, in the worst case, major catastrophes can occur.
- Computer software must be designed to allow flexibility and human choice. Laws and regulations should be interpreted by humans, not by machines.
- Making the software more complex, to include in it more different special handling of special circumstances, will often only make it worse. Instead of complex software, software should be flexible and open-ended.

- There is a human tendency when designing software to want to include in it “proper procedure” and “experience how things should be done”. This tendency can easily produce unusable or unsuitable software.
- Possible exception from the above: Certain security rules, where enforcement is necessary to overcome human weaknesses.

13. REFERENCES

Grip 1974: ADB-system och kommunikation (Data processing and communication). Hermods-studentlitteratur, Lund, Sweden, 1974.

Gordon 1970: P. E. T. - Parent Effectiveness Training, by Thomas Gordon, 1970.

Harris 1969: I'm OK - You're OK, by Thomas A. Harris, 1969.

Hoare 1975: Software Design: a Parable . In Software World, vol. 5, No. 9 &10, 1975.

Martin 1997: Empowering Educators and Parents: Content Advisories for the Internet. By C. Dianne Martin, Proceedings of the ITiCSE ACM conference, June 1997.

Palme 1975: Interactive Software for Humans , by Jacob Palme. An abbreviated version was published in Management Informatics vol. 7(1976) pp 4-16.

Palme 1997: User influence on software design may give less good software .

Palme 2000: A personal history of CMC, by Jacob Palme. Honorary, in Creative Crossroads Electronic Honorary Publication, Dedicated to Yvonne Wærn on Her Retirement.

Seligman 1975: Helplessness: On depression, development and death, by Martin E. P. Seligman, W.H. Freeman, San Francisco 1975.

Vetandets värld 2006: (In Swedish broadcasting, audio): Den mänskliga faktorn som förklaring till olyckshändelser. Vetandets värld, Sveriges Radio, 27 Feb 2006.