Is your upgrade worth it? Process mining can tell.

Michiel van Genuchten, Ronny Mans, Hajo Reijers, Daniel Wismeijer

Keywords: usability, process mining, product metrics

Software vendors typically put out updates and upgrades of their software once or twice a year. New technologies like cloud computing allow vendors upgrading even more frequently [1]. The users are faced with the question whether it is worth to pay for and upgrade to the next version of software. The benefit for the vendor is clear: additional revenue from the upgrade and reduced support efforts for older versions in the field. Some argue that new releases are mainly in the interest of the supplier and not the user [2]. There are indications that users feel overloaded and do not see the benefits to upgrade to the latest version of the software [3]. Individuals and companies delay or even skip upgrade cycles and many use 5 year old versions of software that still does the job.

The software industry does not provide much evidence that it is worthwhile to upgrade to the next release. Authors like Tom Gilb [4] have argued for many years that software vendors should be able to quantify their requirements and benefits of their software. A practical and, preferably, automatic way for doing so has been lacking. Some software suppliers such as Mozilla have started collecting metrics to better understand how their products are being used. Their focus is typically on aspects such as UI elements and memory usage [5] and not on the processes supported by the products and the usage times. We propose to use *process mining* to prove that upgrading to the next release provides quantifiable benefits to the end user. Process mining capitalises on the fact that more and more information about processes is captured in the form of event logs [6, 7]. These events can be used to make processes visible and also to show the benefits of using the next release of a software product. We have three potential beneficiaries in mind: first, the end user who obtains improved insight in his processes. Second, the software supplier who by becoming transparent could be able to convince more users to upgrade to the next release, as well as gaining a better understanding of the way their software is being used. Third, researchers (like us) who will benefit from increased and improved logging capabilities in all kinds of software.

Process mining

By definition, software is running on machines that can log user behavior. Typically, these user behaviors are stored in a so called *event log*. Process mining aims at extracting useful process-related information from event logs. Process mining describes a family of a-posteriori analysis techniques exploiting the information recorded in event logs [7]. For these event logs it is important that each event refers to a

well-defined step in the process (e.g. the saving of a file) and is related to a particular case (e.g. a particular user who is writing a document). Also, additional information such as the performer of the event (e.g. the end user who is writing a document), the timestamp of the event, or data elements recorded along with the event (e.g. the storage location of a file) may be stored.

As illustrated in Figure 1, software systems may support processes, control the work of people and organizations, and may control machines or components. In practice, there is often a significant gap between what is prescribed or supposed to happen, i.e. the model which describes the allowed behavior of a system, and what actually happens. As systems produce event logs and thus record (parts of) the actual behavior, it can be identified what happened in reality.

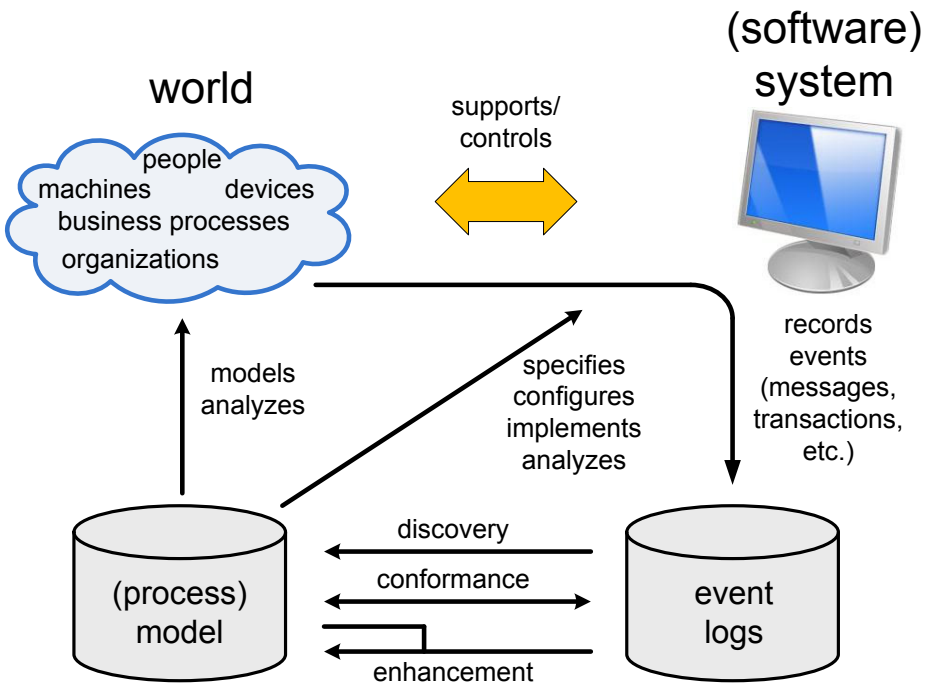

Figure 1: Three types of process mining: (1) discovery, (2) conformance, and (3) enhancement.

As shown in Figure 1, we consider three types of process mining: (1) discovery, (2) conformance, and (3) enhancement. In this paper, we will use it for discovery purposes only. The discovery aspect of process mining aims at inferring process models that are able to reproduce the observed behavior. For example, in the context of a word processor, the discovered model may describe the typical steps taken in order to include a figure in a document or the steps taken for the final layout of a document. The discovery we are interested in is to quantify the benefit of upgrading to the next release of the software.

Applied to digital dentistry: Design of Crowns and Bridges

We applied process mining to digital dentistry as part of a year-long research project that investigated to what extent workflow technologies could be used to make the transition from analogue to digital dentistry more effective [8]. Up until now, dentistry was mostly carried out in the "analogue" world: patient information was recorded on paper, impressions were poured in plaster to create models, and dental prostheses were waxed-up and cast. Digital technologies and software are changing dentistry as we speak: Impressions can now be taken with intra oral scanners, crowns are designed with CAD software and milled or 3D printed in the dentist's office or somewhere else in the world. Millions of lines of software are involved as part of the scanners deployed, the CAD design software used and the networks to connect the players in the value chain.

As with all software, vendors update and upgrade their software regularly and users face the question whether the next release is worth the upgrade. We applied process mining to one of the dental design software packages in the field. The workflow consists roughly of three parts: first a digital scan of the tooth of the patient is produced. A scan can be made directly in the mouth of the patient with an intraoral scanner. Another possibility is to use a traditional impression that is later scanned with a lab scanner. The digital scan file is the input for the second step in the workflow: the design of the dental element (typically a crown or a bridge). The design work in the middle is typically done by a dental technician or digital dental designer, either employed in the dental office, in a dental laboratory or in an off-shore design center somewhere in the world. Typical dental elements designed are crowns (of different kinds), bridges and abutments (the connector between a dental implant and a crown). Output of the design is an STL (STereoLithography) file that can be sent to a milling machine or 3D printer that can manufacture the dental element.

Process mining was applied to the design software in the second step of the workflow. Events that were logged in the software were for example the creation of a new order, scan import done, CAD engine started, CAD engine done and design done. In Figure 2, the control-flow, resource, and performance perspective of the business process followed for designing 751 products is shown. The model shown in the figure is a Petri net [7]; rectangles represent tasks and circles, called places, represent a state in the process. Fully black rectangles are only added in order to accurately describe the flow of work and therefore do not represent a task. Moreover, several places have multiple outgoing arcs. This represents a choice in the process and only one path may be followed. For each outgoing arc, the probability of following the respective path is indicated. The color of a place provides an indication about the average waiting time that is spent in the place. A pink colored place indicates that a high average waiting time exists for the place (more than 60 minutes), a yellow color indicates a medium average waiting time (in between 30 and 60 minutes), and a blue color indicates a low average waiting time (less than 30 minutes). Finally, note that the prefix of a task indicates the role that needs to be fulfilled by the person which is allowed to perform the task. Here, the 'user' role is represented by the 'U' prefix.

The control-flow and the roles for the resource perspective are as follows. First, an order is created (“U:Creation and First Initialization” task). Afterwards, several steps can be taken. Note that in order to perform a certain step, the order first needs to be locked (“U:Locked” task). Afterwards, the order is unlocked again (“U:Unlocked” task). After creation of the order, the model of the patient’s teeth can be scanned (“U:Scan Done” task) or a file of a previous scan can be imported (“U:Scan Import Done” task). Next, there is the option to let the CAD Engine make a proposal for the design of the dental element end product(“U:CAD Engine done” task). Within the software, this always needs to be triggered by a user. Afterwards, the last step is to complete the design (“U:Design Done” task).

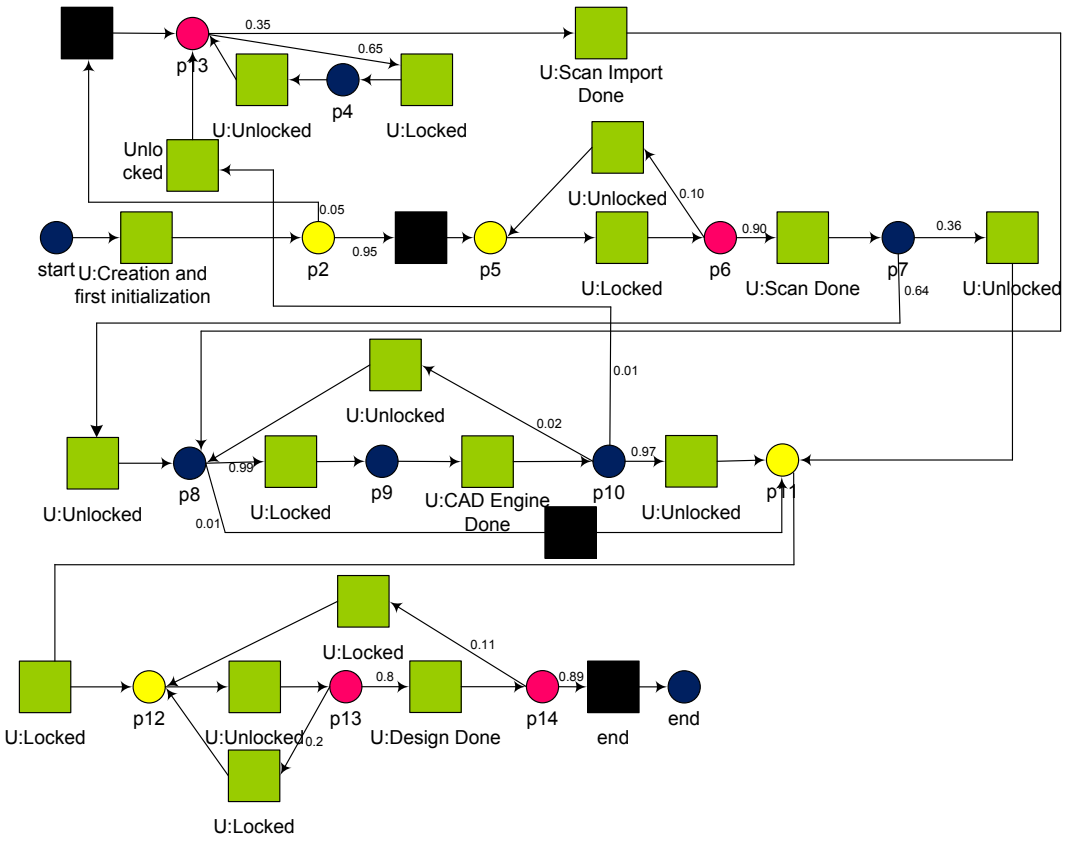

Figure 2: Business process showing how the Dental lab technicians used dental design software for designing various dental products (e.g. a crown or a bridge).

Quantifying the benefit of the next release

Our goal was to quantitatively compare the times users need for basic use cases of two subsequent releases of the software. In other words: is the upgrade worth it for the user. The CAD software in use already logged the start and finish of a new activity. Some additional details were logged to make a

meaningful comparison possible. Some algorithms were reused and developed to extract the logging data from the CAD program. The additional effort can be measured in person days of programming to allow some experiments with experienced customers. We report here on the early application with one dental lab that went as one of the first from an existing to a new release. Using the automatic logging we were able to compare 900 designs made with the old software to 500 designs made with the next release of the software. The results are provided in Figure 3.

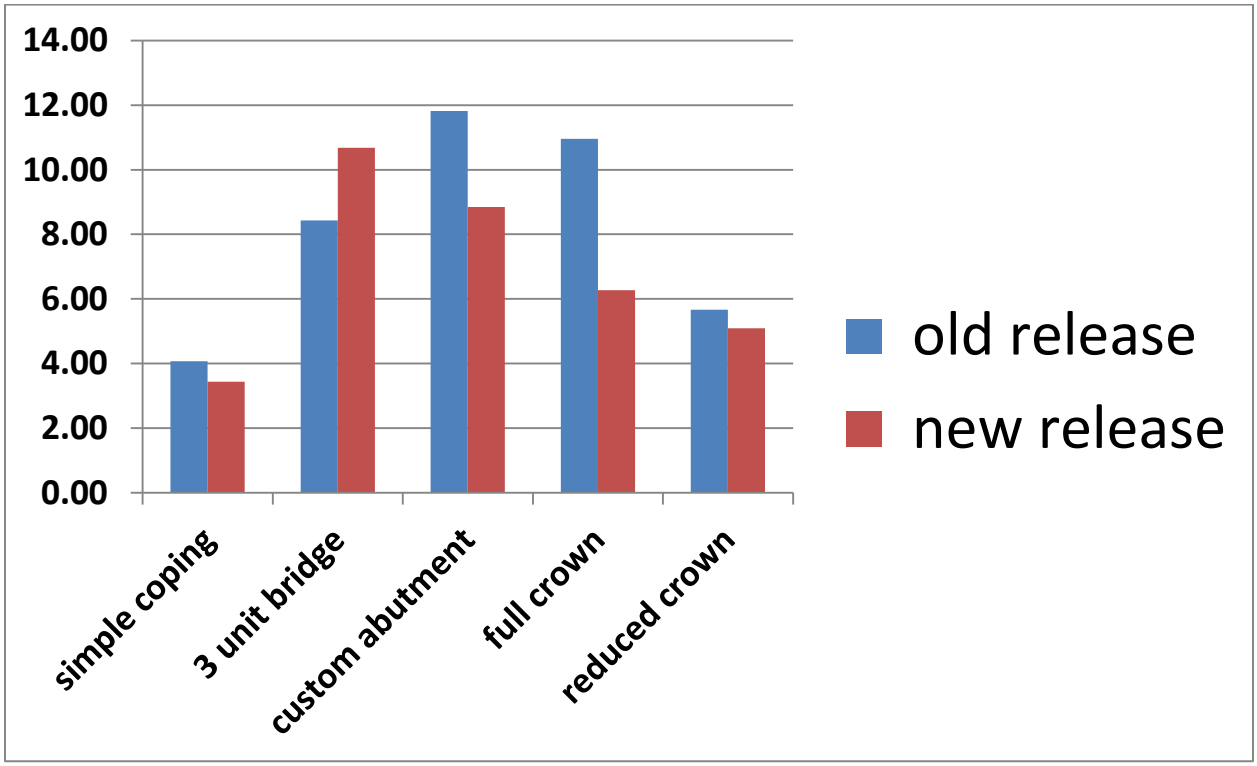

Figure 3 Dental prosthetics designed with the software which are simple coping, 3 unit bridge, custom abutment, full crown and reduced crown. The design time is given on the y-axes in minutes. The average design time with the new release was 11 percent less than with the old release.

All classes represented in figure 3 include at least 20 designs. Other dental indications with fewer than 20 designs were not included in the comparison. In four of the five indications the new software proved to be more efficient. The only exception was the 3-unit bridge where the design took about 20 percent more time. An analysis indicated that this was due to a bug in the early release of the new version that required additional manual positioning of dental elements because the automatic positioning was flawed.

The various stakeholders benefited in different ways from the insights that the process mining analysis provided:

- For the end user, the average design time with the new version was 11 percent less than with the old version of the software. The benefit of the upgrade could easily be quantified in time and money. The upgrade was worth the trouble.
- The impact of bugs became much clearer for the engineering team that developed the software. It was clear that hundreds of customers were losing time every day as long as the bug was still in the field. This allowed a much more quantitative evaluation of the priority of a bug fix versus the development of a new feature.
- The company involved accumulated some quantitative and proven marketing arguments to convince other customers to go to the next release. To be 11 percent more efficient, based on measuring hundreds of cases, is a much better argument than "we advise you to upgrade because it is better for you".

Using process mining for other software products

The approach used can be applied to many other software products. Typically it follows the five steps below.

1. **Define a number of leading use cases:** A number of use cases that are frequently executed should be defined. Some examples are: the steps taken to put a new address in a car navigation system; the activities for creating a new document until the completion of the first page; the steps for scanning the lungs of a patient in an X-ray scanner and the preceding definition of the required settings; and finally, in the context of our dental example, the design of a 3 unit bridge based on an STL file generated by a dental lab scanner. For many software products it should be possible to define a number of simple use cases. For some software products (e.g. an operating system) it may require some more thinking and some standard benchmarks; we will address this point in the next section.

2. **Apply process mining to the logs that are already available for the software:** In most cases some form of logging is already available. For example, the time at which a file is created is often logged, as well as the last time an update is stored or a result is sent. These logs can already be used for obtaining some preliminary results. Furthermore, and probably even more importantly, the results provide insights into interesting questions that can be answered and the steps that need to be recorded in order to answer these questions. For example, in the context of our dental example, some design steps may be recorded in order to learn about the approach that is followed by users in order to design a 3 unit bridge.

3. **Build improved logging in the new version of the software:** As part of the development project, improved logging can be built in. As already indicated in Section 2, the logging requirements for process mining are simple. It is already sufficient that for a step, a corresponding name, the timestamp at which the step occurred, and the case to which it belongs is recorded. If needed,

the performer of a step and additional data attributes (both for a case and an event) can be saved. The extension of the software with these logging capabilities should be a limited effort, measured in software engineering days.

4. **Do a pilot with some experienced users:** Once improved logging is in place, it can be used for various analyses. For example, when beta users have started using an upgrade of the software, its impact can be analyzed. In this way, it can be seen whether the upgrade is really an improvement compared to the software that is currently in the field. Within the process mining field, there are many algorithms and several open source tools available in order to compare and analyse process mining logs (see for example www.processmining.org for the process mining framework ProM or [7]).

5. **Analyse the behavior of all users using the new version of the software:** Several weeks or months after the upgrade of the software, the behavior of all users can be analyzed. This helps to better understand them, as well as to improve their processes. For example, knowledge can be obtained about non-frequently used functionalities or functionalities that are used in a different way than expected. Subsequently, the obtained insights can be used for the requirements of a next version of the software.

Generalization

In what circumstances can process mining be applied to ensure that the upgrade is worth the trouble? Five conditions need to be fulfilled:

- **Technical conditions**: it is a necessity that process data is stored in the device and that the data can be accessed after the software has been put to use. Practically, this means that it must be possible to connect to a device to download the process data. Devices like vacuum cleaners and shavers are excluded for this reason. Software in cars and consumer electronics devices are often not easily accessible after sales. Some successful process mining studies have been done in more sophisticated, connected devices such as medical scanners and expensive industrial equipment. For example, in [9] process mining has been applied to the event logs of more than thousand X-ray machines all over the world, in [10] the test process of wafer scanners has been investigated, and in [11] the instrument usage during a laparascopic surgery has been investigated. Our example shows that it does not have to be a million dollar device to justify an analysis of the actual use of a system.
- **High-frequency processes**: it is important that a sufficient number of cases are available. Use cases with a very low frequency are not attractive candidates for process mining. Mining processes that only occur a few times a year is not recommended.

- **Willingness of the software supplier**: the supplier of the software needs to be prepared to provide more inside into the worth of the upgrade. There is a lot of ongoing research into upgrade strategies in the software industry, see for example [2, 12. We focus on providing a new way to quantify the users benefit of an upgrade and will not summarize the body of literature within the context of this paper. It is clear that the user is not the only stakeholder and that the benefit of the user is not the only variable in the equation. Sometimes it looks like the benefit of the supplier is the main reason for the upgrade, for example to lower the maintenance burden or justify an annual fee. There are several software markets where one or a few suppliers control the market. Transparency may not be a high priority among suppliers in such monopoly or oligopoly. However, we expect that in a market with multiple suppliers proving the value of your upgrade is a path that some companies will choose. This will stimulate others to follow over time.
- **Identifiable and comparable use cases**: in our pilot, the use cases before and after the release were comparable and easy to identify. That may not always be the case. For example, an upgrade of an operating system may have an effect on many different use cases. Comparing them all may be a lot of work and requires to attach a weight to each use case to be able to make an overall comparison. Also, there may be new use cases, such as integration support of the basic functions on a new device, such as a tablet. Two comments about this. First, we think a lot of software does have end user functionalities that can be compared before and after release. Second, more mature industries have developed benchmarks that allow the comparison of a new release to an old release of their product on certain aspects. Take, for example, the fuel consumption that can be compared among subsequent versions of very different cars. In some cases this is done by neutral, third parties to increase the chance for an objective comparison.
- **Approval of the user**: the user needs to agree that the data he/she supplies and is stored in his machine is fed back to the supplier. This is not trivial since there may be different kinds of users involved. For example, in our case there was a dental lab (the early adopter) and medical data was involved. In this case, it has to be ensured that the laws regarding medical data are respected. Therefore, we only used the design times and never touched the medical data. Also, identifiers were anonymized such that no link with any involved patients could be made. Representativeness of the users that are involved is another important issue.

This seems like quite a list of conditions. The question is for which percentage of the software products out there would process mining be applicable? Let's take the products that have been described in the Impact column in IEEE Software since 2010. In total, 14 products have been described, ranging from software in the Airbus 380 [13] to car navigation [14]. An overview is provided in [15]. We are optimistic about being able to apply process mining in a meaningful way to determine the value of an upgrade for all 14 products described so far. In some cases such as car navigation, it could be as straightforward as described in this paper. In other cases, such as the Solaris OS [16], it would require some more thinking, some more patching, and some coding. Yet, the essential conditions seem to be satisfied.

Conclusions

Software suppliers should increase their efforts to convince the users of the necessity of an upgrade. Process mining is instrumental in this, with limited additional effort. The application of process mining in this context will give the user and his evaluation of upgrades a more dominant role in the development process. Two examples of the line of reasoning in such situations: “3D should not be built in the UI just because we can, but because we think that the data will show a reduced effort for the basic use cases for the end user”. Another example: “We will not increase the update frequency of our software to four times a year because cloud technology allows for it. We will not update at all until the data shows that an update is in the benefit of our users”. Those software companies that think this is too much trouble should look at the pre-launch activities that car manufacturers and medical equipment suppliers have to go through before they are even allowed to go to market. Just automatically collecting some user data and providing feedback to the engineering teams and the users is a limited effort compared to this. We expect that, over time, those software companies that are not willing to explain their customers the value of an upgrade will lose market share to those who will.

References

[1] Keizer, G., Microsoft_We_can_update_Office_by_subscription_every_90_days, Computerworld, March, 1, 2013.

[2] Ellison G. and Fudenberg, D. The Neo-Luddite's Lament: Excessive Upgrades in the Software Industry, *The RAND Journal of Economics, 2000,* 31(2):253-272

[3] Rutkowski, A.F., Saunders, C.S., Hatton, L., Generational Impact of software, IEEE Software, May-June, 2013

[4] Gilb, T., Principles of Software Engineering management, Addison Wesley, 1988

[5] Browser metrics, Mozilla, https://wiki.mozilla.org/Browser_Metrics

[6] van der Aalst, W.M.P., Using Process Mining to Bridge the Gap between BI and BPM. IEEE Computer, 44(12):77-80, 2011

[7] van der Aalst, W.M.P., Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer-Verlag, Berlin, 2011.

[8] Mans, R.S., Reijers, H.A., Wismeijer, D., van Genuchten, M., A Process-oriented Methodology for Evaluating the Impact of IT: a Proposal and an Application in Healthcare. Information Systems, 38(8): 1097-1115, 2013.

[9] Günther,C.W., Rozinat, A., van der Aalst, W.M.P., van Uden, K. Monitoring Deployed Application Usage with Process Mining, BPM Center Report BPM-08-11, BPMcenter.org (2008).

[10] Rozinat, A., Jong, I.S.M. de, Gunther, C.W. & Aalst, W.M.P. van der (2009). Process mining applied to the test process of wafer steppers in ASML. IEEE Transactions on Systems, Man and Cybernetics, Part C, Applications and Reviews, 39(4), 474-479.

[11] Blum, T., Padoy, N., Feuner, H., Navab, N., Workow Mining for Visualization and Analysis of Surgeries, International Journal of Computer Assisted Radiology and Surgery 3 (2008) 379{386.

[12] Amit Mehra, Ram Bala, Ramesh Sankaranarayanan, Competitive Behavior-Based Price Discrimination for Software Upgrades, Information Systems Research, Volume 23 Issue 1, March 2012.

[13] Schaminée, H. and Aerts, H. Short and Winding Road: Software in Car Navigation Systems, IEEE Software, vol. 28, no. 4, 2011, pp. 19–21.

[14] Burger, S., Hummel, O., Heinisch, M., Airbus Cabin Software, IEEE Software, vol. 30, no. 1, pp. 21-25, Jan.-Feb. 2013, doi:10.1109/MS.2013.2

[15] van Genuchten, M., Hatton, L., Quantifying software's impact, IEEE Computer, October 2013,.

[16] King, C., Beal, C., CSI Kernel: Finding a Needle in a Multiterabyte Haystack, IEEE Software, vol. 29, no. 6, pp. 9-12, Nov.-Dec. 2012, doi:10.1109/MS.2012.154