Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-3 (2015) 848–854

Service Architecture and Interface Design for Mobile Machine Parameter Optimization System

Petri Kannisto, David Hästbacka,
Matti Vilkko, Seppo Kuikka

Tampere University of Technology, 33101 Tampere, Finland
(e-mail: {petri.kannisto|david.hastbacka|matti.vilkko|seppo.kuikka}@tut.fi)

Abstract: Performance improvement is a constantly important topic in manufacturing, and mobile work machines are no exception. In some mobile machines, machine behaviour is affected by various device parameters. Their effect can be examined by statistical analysis, but exploiting analysis results during machine operation requires a sophisticated distributed information system. This paper introduces an architecture for such a system suitable for geographically dispersed machine fleets. It covers aspects on data collection, storage, analysis and utilization. Once statistical analysis has been performed, its results are made available for applications that assess machine performance locally during operation. If there is room for improvement in parameter values, the operator of the machine is given suggestions to change them. A prototype implementation is presented. The results show that such an information system has a considerable potential of bringing competitive advantage to machine operators. Besides, added value is also expected to the manufacturer of the machine as performance-related knowledge is augmented. Thus, further service business development is also contributed.

Keywords: Service Oriented Computing, In-process Manufacturing Monitoring, Manufacturing Resources and Processes

1. INTRODUCTION

In the current global business climate, enterprises are constantly searching for advantage over their competitors. Saving time and costs in any part of a production chain will improve overall performance and productivity. While productivity is present in any type of work, mobile work machines are essential in goods production.

The productivity of a mobile machine is not only dependent on the operator but also on how the machine itself performs. The performance of some mobile machine can be tuned by adjusting various machine parameters. Typically, a mobile machine consists of several subsystems. The scope of optimization is not only a single subsystem but the overall productivity of the machine. The number of parameters can be high.

Despite the importance of parameter tuning from the performance point of view, it is usually performed by machine operators that might not have a solid understanding of how each parameter affects machine behaviour. While a domain expert might know quite well what kind of parameter values are typically good, the information is not available to everybody. That is, there is a need for means to bring domain knowledge available for various actors. Besides, even the most skilled domain expert does not know how all possible parameter value combinations affect. Current knowledge is typically related to adjusting a limited number of parameters. Thus, there is an obvious need to gain new knowledge.

A lot of parameter tuning knowledge is present in performance data. To exploit the data, statistical methods can be utilised; this would produce new information about how parameter changes affect machine performance. A lot of data is readily available: a modern mobile machine has countless sensors in its subsystems that can be utilized for performance calculation. In addition, exposing parameter values for computation is only a matter of systems integration.

The scope of this paper is what kind of information system architecture is required to meet these requirements. Data should be retrieved, stored and analysed to gain new knowledge. Then, the knowledge should be stored and utilised for parameter optimisation in machines in their current operational context to improve performance.

While improving machine performance is an obvious competitive advantage for the operating company, the new knowledge is beneficial for machine manufacturer as well. The system is an extensive source of machine information, enabling better machine design in the future. That is, new added value will be gained from existing machine measurement and information systems.

The structure of the paper is as follows. Chapter 2 describes the research context. Chapter 3 gives the requirements of the system that is the basis of this paper while chapter 4 explains the architecture meeting the requirements. Chapter 5 provides a description of the current prototype, and finally, chapter 6 contains results and discussion. The entire paper is concluded in chapter 7.

2. RESEARCH CONTEXT

2.1 Research Methodology

In this work, *constructive design science research* has been applied as the research methodology. The idea is to develop new software-related artefacts (or constructs) iteratively so that new ideas are discovered. Combining the points of view from constructive research and design science research have been studied by Piirainen & Gonzalez (2013).

This paper applies the methodology for designing a conceptual system architecture. The requirements of the system are considered, and a solution is proposed so the various aspects are met.

2.2 Related Work

The core areas of this work include generic system interfaces, mobile machine fleet data collection and mobile machine related expert systems. Related publications are summarised in this chapter.

Gelgele & Wang (1996) introduce an expert system for automobile engine fault diagnostics. Its purpose is to assist mechanics in their work.

Lu et al. (2000) have designed an expert system for automotive fault diagnostics. The system generates new knowledge by self-learning. The rules in the expert system are based on fuzzy values.

Kher et al. (2001) propose utilising a neural network for vehicle diagnostics. A fault table is utilized to recognise various faults.

Saxena et al. (2005) have utilised data collected from vehicles to analyse the symptoms of defects so vehicle maintenance can be assisted. Their method utilises case-based reasoning. Further, information is extracted from textual reports written by human workers. The idea of the paper is analogous to this as data from vehicles fleets is processed.

Dingus et al. (2006) documents a 100-car driving study. Its purpose was to collect vehicle data in everyday conditions so different analyses can be performed about driver behaviour and various incidents.

Mackiewicz (2006) introduces a generic interface for power system applications integration. The idea is to facilitate integration with business-related information systems. Web Services are utilised to reduce the amount of integration work, and a common data exchange model is also used to promote flexibility.

Ribeiro et al. (2008) have designed a generic interface definition for delivering messages between the devices of a manufacturing system. The intention is to reduce the amount of work required for system changes. DPWS (Devices Profile for Web Services) is utilized as the foundation technology; it is a Web Service framework targeted especially for low-resource devices.

Wu et al. (2008) propose a system for combustion engine diagnostics. First, variations in engine sound output are utilised to recognise exceptional situations. Then, a neural network based expert system is proposed. In a following study (Wu & Liu 2009), further signal processing methods are researched.

Midlam-Mohler et al. (2009) have studied data collection from Plug-In Hybrid Electric Vehicles (PHEV). The focus is to collect real-world data about charging and duty cycles.

Kannisto et al. (2014) have examined the aspects of mobile machine operator performance assessment. Their system measures overall machine performance and generates feedback for operators. The required domain knowledge is modelled and utilized as rules.

Hästbacka et al. (2014) have worked to create a generic data model for integrating automation equipment data. While the context of the paper is industrial automation, the idea is generalizable to other uses as well.

None of the references papers, however, have considered how aggregated and analysed information can be provided back to the vehicle or mobile machine. There seems to be no published research on such information provision – e.g. in order for a vehicle or a mobile machine to have information available that could improve its performance and productivity.

3. PROBLEM DEFINITION AND SYSTEM REQUIREMENTS

The ultimate goal of the system is to maximise the performance of a mobile machine by utilising performance data and domain knowledge to provide parameter feedback (see Fig. 1). The context where the machine is operating must also be considered. From this main objective, several requirements can be derived.

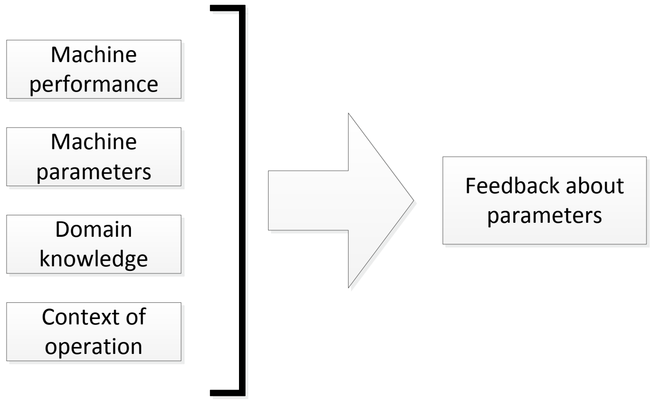

Fig. 1. The core idea of the application illustrated.

To get maximal reliability and coverage of knowledge, data must be gathered from multiple machines. It must be possible to run data analyses in an office, but their results must be accessible for onboard performance analysis in machines.

To get new domain knowledge, statistical analysis is applied to data – while statistical analysis methods are out of the scope of this paper, they are crucially important anyway. The domain knowledge is stored and maintained through a user interface that must be easy enough to be used without any special ICT skills. This is because domain experts are rarely ICT experts and vice versa.

Another need for statistical analysis is getting aggregated performance and parameter values related to various machine types. As the working contexts have multiple dimensions – terrain, weather, the characteristics of the material being processed, and so on – the methods must be sophisticated enough so the dimensions can be recognized and considered in the analysis. What might be a reasonable performance value in some conditions might be low in another. The same applies to the values of parameters. Further, similar machine behaviour may not be expected in all the purposes for which the machine is utilised. Even though statistical methods themselves are not considered here, the system must enable storing their outcome so any relevant context properties can be taken into account in performance assessment.

If some machine parameter needs tuning, the machine operator should get feedback about it while the machine is being operated. Multiple parameters may affect a single performance indicator. Of those parameters, it should be considered which parameter has the most significant effect and suggest its adjustment first. Once it is in the proposed range, adjusting the second most significant parameter is suggested and so forth.

Feedback should only be given after a few work cycles have been finished. This is because the effect of random factors may be too high if only a single work cycle is considered. In addition, some machine parts, such as certain hydraulic components, might not reach their best performance right after machine startup but only after a short period of operation. Further, it might be necessary to evaluate the results of each committed parameter change as they could actually decrease performance. In such a case, reverting parameter changes may be appropriate.

Mobility sets a considerable restriction to Internet connectivity. A mobile machine cannot be assumed to have a stable Internet connection all the time. For this reason, data caching is necessary. This applies not only to the data collected from an individual machine but also to domain knowledge as well as aggregated parameter and performance values stored from data analysis results.

Considering the lifecycle of the system, both knowledge and data maintenance must be flexible. This is because the outcome of statistical analysis as well as knowledge is expected to improve. The system must not limit introducing new aggregated values or knowledge units or even removing or replacing existing ones. That is, even if there were changes in data and knowledge items, the system must always remain functional even though its outcome (i.e. feedback) may change as the result.

In knowledge representation, fuzzy values must be utilised instead of bare numbers. In this paper, fuzzy refers to the classification of numeric values so each class expresses the interpretation of various values in their context. For example, for a certain machine type in typical conditions, if a work phase lasts 10 seconds, it could be classified as an average performance. In more demanding conditions, 10 seconds could indicate even a good performance. Similarly, getting 100 units of output might be low, average or good depending on the context.

Utilising fuzzy values has multiple advantages. First, a domain expert cannot be expected to have a comprehensive understanding of what kind of numeric performance measures and parameter values are low, average or high for a certain machine type. Second, while numeric performance and parameter values vary between machine types, similar knowledge units (i.e. rules) may treat corresponding values anyway. That is, if numeric context information is separated from rules, they have a considerable reuse potential among machine types. E.g. the "high" unit output value factor used in rules is different depending on the machine type, accessories, geographical area and the work task at hand. This, however, requires a large fleet with data so that sufficient statistical analysis can be performed.

4. ARCHITECTURE

Considering the requirements set in the previous chapter, this chapter describes the system architecture and its components. The architecture is illustrated in Fig. 2, and each part is explained in the following sections.

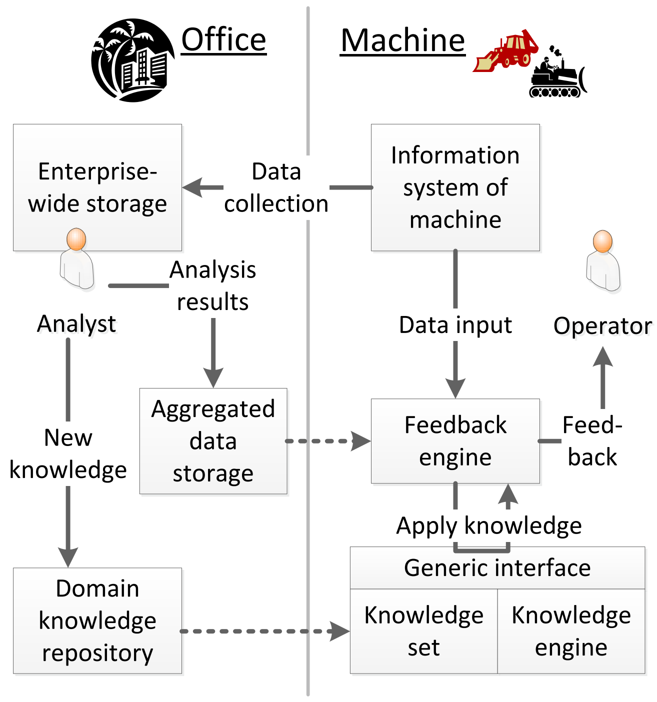

Fig. 2. Architecture illustrated.

4.1 Overview

The most predominant requirement is that while some actions are performed in an office, others are performed in each machine instance. As a result, there is a strict division in offboard (office) and onboard (machine) functions. While all the components are important for the system, this paper is mostly focused on the components run in machines, leaving the office side (and the statistical analyses) as a future topic.

The ultimate source of information is the *information system* in each machine. They provide both parameter values and measurements, the latter of which may be further processed to indicate machine performance. Data is supplied for two components: the *enterprise-wide data storage* of the machine manufacturer and the *feedback engine* that generates feedback for machine operator.

The enterprise-wide data storage is the basis of data management. It shall contain a wide collection of data from multiple machines operated in various conditions. The data is made available for further processing. During the lifecycle of the system, more data is collected continuously. In addition, different statistical methods are applied to the data so new knowledge is gained.

The *aggregated data storage* shall store results from statistical analysis. Analysis results are organised after machine types and various context properties such as terrain and other conditions. This data is available for the feedback engine so performance and parameter value assessment can be performed.

The *domain knowledge repository* is another consumer for analysis results. While the aggregated data storage contains numeric values, domain knowledge is more sophisticated and less concrete. It stores the knowledge of domain experts – it is edited manually with a user interface that does not require any special ICT expertise from its user. Even the majority of its contents may come from the existing knowledge of domain experts without the assistance of statistical analyses, but statistical methods are expected to contribute to it anyway. The concrete implementation of the domain knowledge repository is a rule modelling system. The rules must be both human-readable and machine-processable.

The coupling of office side and machine side must be minimal. This is not only because machines might have no persistent Internet connection but also because the machine side operates continuously whereas the office side is likely to operate periodically. That is, both the contents of the aggregated data storage and the domain knowledge repository must be packaged so that their distribution to machines is straightforward. Due to the assumption of long periods without Internet connectivity, a local copy in each machine is a necessity. However, the local copy should be refreshed whenever possible.

On the machine side, a *knowledge engine* is executed to utilise the domain knowledge modelled on the office side. While the engine is an executable component, the actual knowledge is a transferable rule set. The knowledge engine has a generic interface wrapping it so it is easy to replace the entire knowledge engine if it is necessary.

The feedback engine generates the ultimate output of the system: feedback for the machine operator. It shall apply domain knowledge and aggregated data to analyse local performance measurements and local parameter values to run the actual assessment.

4.2 Design Principles of Interfaces

Each component interface shall be as generic as possible. Loose coupling is an essential principle: when dependencies between components are minimal, it also minimises the work required if some component is modified, upgraded or even replaced. Moreover, loose coupling enables component distribution over a network. Finally, component interfaces should be platform-independent so any technology can be utilized to implement a component. These requirements are met by following the principles of service-orientation and by choosing appropriate technologies for implementation.

Further, the data model of component interfaces should be generic as well. The key is not to model each type and property as such but rather have a generic data model so any type with any property set can be represented. This way, the principle of loose coupling is taken even further. The core of the model is that each object has a class name, an object ID and an arbitrary number of properties. A property can be a simple atomic property or a reference to another object (that is, an object ID) – references enable even complex object hierarchies. Such a data model for object representation is suitable for invoking a rule engine. A drawback of the loosely coupled object model is that the object model itself cannot define the schema of the data. While the schema is loose, object instance validation must be performed elsewhere. However, while the validation operation may cause extra work, the advantage of loose coupling is to reduce re-engineering work and turn it towards reconfiguration during the lifecycle of the software.

To have a generic service interface, existing specifications such as Resource Description Framework RDF (see Beckett & McBride 2004) could be utilised. As RDF data model and related techniques support data processing with a machine, RDF could bring added value – however, considering the current requirements, a light and simple data model is enough so the additional complexity brought by RDF is not desirable. Besides, only little work is required for designing a simple generic object schema.

5. PARAMETER FEEDBACK PROTOTYPE

With the architectural model of the previous chapter, a prototype can be implemented. As several components must be implemented to meet the various aspects and requirements, the first prototype only covers certain points of view.

The prototype is illustrated in Fig. 3. The only application on the office side is Drools Workbench. On the machine side, a feedback engine utilises other components to get the ultimate output. Each component is explained in the coming paragraphs. In general, loose coupling is followed by utilizing XML-over-HTTP (also called Representative State Transfer, REST or "restful") interfaces between components. Both XML and HTTP are platform-independent techniques so they bring a high degree of flexibility in terms of replacing or upgrading system components. Further, generic interface design encapsulates underlying implementations.

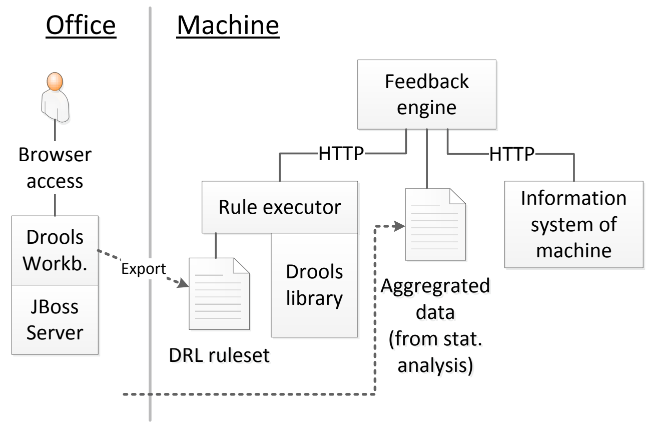

Fig. 3. Prototype illustrated.

Domain knowledge determines how analysis results are compared to the performance and parameter data of an individual machine. Domain knowledge is modelled as rules using Drools Workbench rule framework run as a server application in an office. Drools Workbench has a browser interface, and it enables rule modelling with decision tables. Decision tables are an intuitive tool even for persons not familiar with textual rule modelling languages but their drawback is their relatively limited schema. Once rules have been modelled, they are exported as DRL (Drools Rule Language) files and delivered to machines for utilisation.

Feedback engine implements the core workflow of the system (Fig. 4). It retrieves data from the machine information system and feeds it to the rule engine. Most performance values cannot be measured as such: for example, calculation of averages over various values is typically required. Further, as rules in this application do not refer to raw numeric performance and parameter values, they must be detached from their context by transforming them to corresponding fuzzy representations. Then, they can be fed to the rule engine. To generate fuzzy values, aggregated results from the statistical analysis are necessary. Once rules have been invoked, their results are shown to the operator.

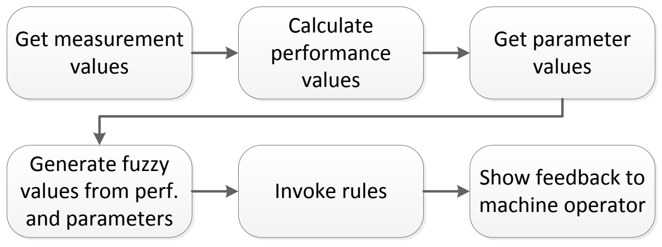
Get measurement values
Calculate performance values
Get parameter values
Generate fuzzy values from perf. and parameters
Invoke rules
Show feedback to machine operator

Fig. 4. The workflow of the feedback engine.

The rule executor has been wrapped with a generic XML-over-HTTP interface. It has been implemented with the Drools rule engine component. To reach maximal flexibility, the schema of the rule executor interface models all objects similarly regardless of their class or property set. This is reached by having a generic schema that allows an object to have any number of properties.

Aggregated data is currently hard-coded in the application. It contains statistical distributions generated from machine data gathered from a large fleet over a long period of time, enabling the comparison of a prevailing parameter and measurement value set. In the future, there should be a module that holds a local cache of the aggregated data but tries to retrieve a new one regularly so the data can be refreshed.

The information system of the machine provides parameter and measurement data for analyses. It has readily an XML-over-HTTP interface so no wrapper layers need to be utilised in its integration to the restful architecture. In addition, the schema of the interface is flexible and adaptable so the principle of loose coupling is fulfilled.

The functionality of the prototype can be demonstrated with a concrete example. Let us consider the following scenario: (1) operator productivity seems subpar to statistical average productivity in similar conditions, and (2) the operator does several corrective manoeuvres on the hydraulic boom of the machine. In such a case, the system suggests lowering the speed of the hydraulic boom by decreasing the flow of hydraulic fluid to the cylinders of the boom. This could be achieved, for example, by restricting the flow using valves.

The rule given in the example is illustrated in Fig. 5 as a decision table row. In the row, three variable values are examined. For example, for V1, the rule framework would treat the conditions as follows: "if there is a variable called 'productivity' and its value is *lower than average*". If all the conditions of the rule are met, the following action will be suggested: "*lower* the value of variable 'fluid to boom'". Drools rule engine will generate a DRL expression from each decision table row so they can be executed in an inference process.

Conditions (IF)						Actions (THEN)	
V1 name	V1 value	V2 name	V2 value	V3 name	V3 value	Action	Action target
Productivity	Lower than avg	Corrective manoeuvres	Higher than avg	Fluid to boom	High	Lower	Fluid to boom

Fig. 5. Rule about lowering the amount of fluid directed to boom.

A considerable amount of preprocessing is required before a rule having such a high abstraction level can be invoked. "Fluid to boom" is a machine parameter that controls the opening of a hydraulic valve directing flow to the boom. In contrast, “productivity” and “corrective manoeuvres” are derived variables that must be calculated from measurements. In addition, the values must be scaled using a statistical distribution; whether a value is “low”, “high”, “lower than average” and so forth is determined in relation to past work cycles of the same machine type. As the rules are executed in machines during their operation, these distributions must be available at runtime. Finally, to receive reliable results from rules, the deviation between work cycles is minimized by calculating an average from a certain period.

To further demonstrate the design of the prototype, let us look at the generic interface of the rule executor service. The classes that appear in rules will not appear as classes in the interface. Rather, all the objects have a common class with an arbitrary property set. In the data model of the interface, the “productivity” variable would be expressed as:

- Object type = “**Variable**”
    - Property name = “**name**” value = “**Productivity**”
    - Property name = “**value**” value = “**lower than avg**”

6. RESULTS AND DISCUSSION

In architectural design, the loose coupling of subsystems as well as data and knowledge was considered paramount. The rule modelling method utilised for domain knowledge modelling is flexible. Changing the rule set by modifying, adding or removing rules does not require any software compiling but simply exporting the rule set from the rule modeller and copying it to machines. An updated version of the rule set could be delivered, for example, when the machine receives other software updates.

The set of aggregated data received as the result of statistical analysis has flexibility benefits similar to the rule set. Whenever the analysis is run with fresh measurement data from machines or whenever new analysis results appear, the result set is simply transferred to machines. Even removing some existing variables from the result set does not break anything. A missing variable may prevent some rule from firing and a new variable may not be utilized until a rule considering it as a factor is introduced, but the system will remain functional anyway.

Finally, the entire rule engine has a generic XML-over-HTTP interface. As the data model of the interface has no direct dependencies to the implementing components, any of them could be replaced with minimal reconfiguration.

Rule modelling is also easy in the prototype. The graphical browser interface of Drools Workbench enables straightforward rule modelling as decision tables even for non-ICT experts. This makes rule set maintenance considerably faster as there is no need for an ICT expert to apply the actual rule change – domain experts can commit changes themselves. Further, as there is no need to communicate rule set contents from domain experts to ICT experts, fewer errors are expected in rules due to potential human communication shortcomings.

However, the modelling methods provided by Drools have also drawbacks. Even though the power of expression of underlying Drools Rule Language exceeds the requirements, decision tables set some restrictions to the structure a rule can have. Were there any complex conditions in rules, a more adaptable method should be utilised. As such, Drools does not provide any other method suitable for non-ICT persons than decision tables. That is, there could be a need to adopt another graphical rule modelling method such as graphs or flowcharts. A modelling tool utilizing them should be implemented for Drools. However, while decision tables are utilised, most of their drawbacks can be overcome by simply having multiple places for conditions or breaking up rules in multiple chained decision table rules.

Further, in the current prototype, aggregated statistical result sets and rule sets must both be delivered manually to machines. Human work is required; besides, the opportunities for a human worker to deliver the files might be few and seldom. To solve this issue, the information should be downloaded by the analysis software over the Internet whenever there is a possibility for it. In the future, the corporate infrastructure of the machine manufacturer should provide means for providing both rule sets and aggregated statistical results for download.

Each machine individual may be slightly different. This is because the manufacturing of machine components is never completely deterministic. For example, each individual electrically controlled hydraulic valve has a different characteristic curve from input current to opening. The current approach does not address this issue as this would require more analysis to distinguish individual variances.

The ultimate goal is to reduce the need for the human operator to adjust parameters. Once there are proper methods to measure machine performance and to optimize it through parameter adjustment, the entire chain could be automated. Then, a better effect of parameter changes is expected. However, automating such a functionality is not trivial so further research is required. It is believed that this method brings most benefit to those circumstances where little thought is put into tuning the machine. Less room for improvement is expected in cases where highly skilled operators fine-tune the parameters.

7. CONCLUSIONS

This article introduces a system architecture for the performance optimisation of a mobile machine. The idea is to collect parameter and measurement data from a large fleet of machines, to store the data centrally, and, finally, analyse it to gain new knowledge. The knowledge is stored in two computationally utilizable formats: *distributions* for pure aggregated data and *rules* for the domain knowledge modelled by humans. Once distributions and rules have been created, they are transferred back to machines in the field so they can be utilized for parameter value assessment in the machines to maximize machine performance. Whenever desired, a new statistical analysis can be performed to refresh the existing knowledge with new data or new statistical methods. An important aspect in rules is that their maintenance shall not require any specific ICT skills such as programming.

In each machine, there is a local information system to utilize the generated knowledge. Machine performance is calculated from measurements and compared to statistical distributions utilizing the rules that hold domain knowledge. If there is room for improvement, the system will suggest the operator of the machine to perform parameter value changes to yield a higher productivity.

Based on the conceptual architecture, a prototype is introduced. Loose coupling is promoted in system interfaces to reduce the amount of re-engineering whenever there is a need to change a system component. Also the fleet-wide statistical factors and the contextual measurements are similarly decoupled from the inference rules providing the parameter change suggestions. This is reached by utilizing fuzzy values rather than bare numerals in rules. The highly distributed nature of the system is considered even in the prototype: the resources retrieved from the office are stored locally as no constant Internet connectivity can be assumed.

Considering the potential of the system, it solves the problems related to parameter value adjustment quite well. However, there is room for future research. The prototype does not meet all the introduced architectural aspects as such. In addition, the degree of automation should be raised in parameter adjustment so the system would perform suggested value changes with no actions required from the human operator.

REFERENCES

Beckett, D. and McBride, B. (2004). RDF/XML Syntax Specification. W3C Recommendation 10 February 2004.

Dingus, T., Klauer, S., Neale, V., Petersen, A., Lee, S., Sudweeks, J., Perez, M., Hankey, J., Ramsey, D., Gupta, S., Bucher, C., Doerzaph, Z., Jermeland, J., and Knipling, R. (2006). *The 100-Car Naturalistic Driving Study; Phase II – Results of the 100-Car Field Experiment.* U.S. Department of Transportation. Springfield, VA, USA.

Gelgele, H. and Wang, K. (1996). An Expert System for Engine Fault Diagnosis: Development and Application. *Journal of Intelligent Manufacturing*, volume 9, 539–545.

Hästbacka, D., Barna, L., Karaila, M., Liang, Y, Tuominen, P. and Kuikka, S. 2014. Device Status Information Service Architecture for Condition Monitoring using OPC UA. In *19th IEEE International Conference on Emerging Technologies and Factory Automation* (ETFA 2014), Barcelona, Spain, 16–19 September, 2014.

Kannisto, P., Hästbacka, D., Palmroth, L. and Kuikka, S. (2014). Knowledge Management Architecture and Rule Based Reasoning for Mobile Machine Operator Performance Assessment. In *16th International Conference on Enterprise Information Systems* (ICEIS 2014), Lisbon, Portugal, 27–30 April, 2014.

Kher, S., Chande, P.K. and Sharma, P.C. (2001). Automobile Engine Fault Diagnosis Using Neural Network. In *2001 IEEE Intelligent Transportation Systems Conference*, Oakland, CA, USA, 25–29 August, 2001.

Lu, Y., Chen, T.Q. and Hamilton, B. (2000). A Fuzzy System for Automotive Fault Diagnosis: Fast Rule Generation and Self-Tuning. *IEEE Transactions on Vehicular Technology*, volume 49, no. 2, 651–660.

Mackiewicz, R. (2006). The Benefits of Standardized Web Services Based on the IEC 61970 Generic Interface Definition for Electric Utility Control Center Application Integration. In *Power Systems Conference and Exposition 2006* (PSCE '06), Atlanta, GA, USA, 29 October–1 November, 2006.

Midlam-Mohler, S., Ewing, S., Marano, V., Guezennec, Y. and Rizzoni, G. (2009). PHEV Fleet Data Collection and Analysis. In *Vehicle Power and Propulsion Conference 2009* (VPPC '09), Dearborn, MI, USA, 7–10 September 2009.

Piirainen, K. and Gonzalez, R. (2013). Seeking Constructive Synergy - Design Science and the Constructive Research Approach. *Lecture Notes in Computer Science*, volume 7939, 59–72.

Ribeiro, L., Barata, J., Colombo, A. and Jammes, F. (2008). A Generic Communication Interface for DPWS-based Web Services. In *the IEEE International Conference on Industrial Informatics* (INDIN 2008), Daejeon, Korea, 13–16 July, 2008.

Saxena, A., Wu, B. and Vachtsevanos, G. (2005). Integrated Diagnosis and Prognosis Architecture for Fleet Vehicles Using Dynamic Case-Based Reasoning. In *2005 IEEE Autotestcon*, Orlando, FL, USA, 26–29 September, 2005.

Wu, J.D., Chiang, P.H., Chang, Y.W. and Shiao, Y.J. (2008). An Expert System for Fault Diagnosis in Internal Combustion Engines Using Probability Neural Network. *Expert Systems with Applications*, volume 34, 2704–2713.

Wu, J.D. and Liu, C.H. (2009). An Expert System for Fault Diagnosis in Internal Combustion Engines Using Wavelet Packet Transform and Neural Network. *Expert Systems with Applications*, volume 36, 4278–4286.