A Lean Automotive E/E-System Design Approach with Open Toolbox Access

Harald Sporer(✉)

Institute of Technical Informatics,
Graz University of Technology, Inffeldgasse 16/1, 8010 Graz, Austria
sporer@tugraz.at
http://www.iti.tugraz.at/

Abstract. Replacing former pure mechanical functionalities by mechatronics-based solutions, introducing new propulsion technologies, and connecting cars to their environment are only a few reasons for the still growing electrical and electronic systems (E/E-Systems) complexity at modern passenger cars. Smart methodologies and processes are necessary during the development life cycle to master the related challenges successfully. One of the key issues is to have an adequate environment for creating architectural system designs, and linking them to other development artifacts. In this paper, a novel model-based domain-specific language for embedded mechatronics-based systems, with focus on the support of different automotive sub-domains, is presented. With the described methodology, the domain-specific modeling (DSM) approach can be adapted to the needs of the respective company or project easily. Though, the model-based language definition can be implemented using various platforms (e.g. Eclipse Modeling Framework), also a custom-made open source editor supporting the DSM technique, is presented.

Keywords: System architectural design · Domain-specific modeling · Automotive embedded systems · E/E-Systems

1 Introduction

The electrical and/or electronic systems (E/E-Systems) in the automotive domain have been getting more and more complex over the past decades. New functionality, mainly realized through embedded E/E-Systems, as well as the growing connectivity (*Car2X-Communication*), will keep this trend alive in the upcoming years. Well-defined development processes are crucial to manage this complexity and to achieve high quality products. Wide-spread standards and regulations, like *Automotive SPICE*® and *ISO 26262*, give a guidance through the development life cycle.

Best practice for the E/E-System development process is still to refer to some kind of the V-Model. Starting with an initialization and analyzing phase, via the subdivided system elements, down to the implementation, and back up by integration and test phases towards the completed system, a multitude of

R.V. O'Connor et al. (Eds.): EuroSPI 2015, CCIS 543, pp. 41–50, 2015.
DOI: 10.1007/978-3-319-24647-5_4

work products arises and have to be managed properly. Trying to keep all of the artifacts consistent manually, is an error-prone and tedious task. Therefore, a lot of effort has been made through the last years to increase the quality by an adequate and highly automated tool support.

To create the system design, most of the existing approaches utilize some kind of UML profile (e.g. SysML[1]). Though these techniques have a lot of advantages, in some scenarios they are not best choice. On the one hand, the possibility to include mechanical parts or the flow of fluids and forces is missing, and on the other hand, a possible lack of UML skills, especially in small project teams, which wants to carry out a lean development, makes the UML-based design to an awkward task.

The main goal of this work is to contribute to the improvement of the existing system architectural design methods. The herein presented approach has been created for the development of embedded mechatronics-based E/E-Systems in the automotive field mainly. However, the techniques are also suitable for other domains. The mentioned improvement is accomplished by extending the widespread and common UML-based methods by domain-specific modeling (DSM) techniques. It's crucial to state that the existing design techniques shall not be replaced by the presented work.

Similar to the previous mentioned de facto standard *Automotive SPICE*, full traceability and consistency between the development artifacts are also one of the main objectives of this work. Various types of requirements are linked to the system architectural design elements, and in the case of requirement changes the affected system parts can be determined easily. Moreover, supported by the DSM definition, a software architectural design can be either created within the same environment as the system design, or a established seamless tool chain can be facilitated after a domain-specific model to UML-based model transformation. In both cases, a Simulink®[2] software framework model can be generated from the software architectural design.

In this contribution the highlighted aspect of the novel DSM approach is the methodology of creating new modeling toolboxes for a particular project or company. The definition of the domain-specific language in combination with the support from the newly developed designer tool, allows a straight forward and intuitive procedure for customizing the DSM to the specific needs. A big advantage of this solution is that the customization can be conducted by the user easily and without coding.

In the course of this document, Section 2 presents an overview of the related approaches, as well as of domain-specific modeling. In Section 3, a detailed description of the proposed modeling approach with a focus on the tailoring for specific (sub-)domains is provided. An application of the described methodology is presented in Section 4. Finally, this paper is concluded with an overview of the presented work in Section 5.

[1] http://www.omgsysml.org/
[2] http://www.mathworks.com/products/simulink/

2 Related Work

In recent years, a lot of effort has been made to improve the model-based automotive E/E-System design methods and techniques. Nowadays, the advantages of a model-based approach are clear and without controversy. Meseguer [12] grants much more reliability, reusability, automatisation, and cost effectiveness to software that is developed with modeling languages. However, model transformation within or also across different languages is crucial to achieve all these benefits.

Traceability, as well as consistency, between the development artifacts has always been an important topic. However, due to the increasing number of electronic- and electric-based functionality, these properties have become vital. If it comes to safety-critical functionalities, according to the 2011 released international standard ISO 26262 [8], traceability between the relevant artifacts is mandatory. A description of the common deliverables along an automotive E/E-System development, and a corresponding process reference model is presented by the de facto standard Automotive SPICE [2]. Neither the functional safety standard nor the process reference model enforces a specific methodology, how the development artifacts have to be created or linked to each other. However, connecting the various work products manually is a tedious and error-prone task.

One of the early work products along the engineering process, is the architectural system design. In the field of automotive E/E-System development, a wide-spread and common approach is to utilize a UML-based technique for this design, like the UML2 profile SysML. Andrianarison and Piques [1], Boldt [3], and many other publications (e.g. [6], [9], [13]) present their SysML methodologies for the system design.

To agree with Broy et al. [4], the drawbacks of the UML-based design are still the low degree of formalization, and the lack of technical agreement regarding the proprietary model formats and interfaces. The numerous possibilities of how to customize the UML diagrams, to get a language for embedded system design, drive these drawbacks. On the one hand, the meta model can be extended, and on the other hand, a profile can be defined [13]. Even if there is a agreement to utilize a common UML profile like SysML, a plenty of design artifact variations are feasible. This scenario doesn't provide an optimal base for the engineer who has to design the embedded automotive system from a mechatronics point of view. Ideally, the tool should be intuitive and easily operated also without specific UML knowledge. These findings led the author to the idea to create a more tailored model-based language for the stated domain. The definition and other details of this language can be found at [16].

Mernik et al. [11] describe a domain-specific language as a language that is tailored to the specific application domain. Enhanced by this tailoring, substantial gains in expressiveness and ease of use, compared to general-purpose languages, should be given. Even if a gain regarding the expressiveness is achieved by the utilization of SysML-based modeling techniques, the ease of use regarding an embedded automotive mechatronics system design is out of sight.

Preschern et al. [14] claim that DSLs help to decrease system development costs by providing developers with an effective way to construct systems for a specific domain. The benefit in terms of a more effective development has to be higher than the investment for creating or establishing a DSL at a company or department. Supplementary, the authors argue that in the next years the mentioned DSL development cost will decrease significantly, due to new tools supporting the language creation like the Eclipse-based *Sirius*[3].

Vujović et al. [17] present a model-driven engineering approach to create a domain-specific modeling (DSM). Sirius is the framework for developing a new DSM, respectively the DSM graphical modeling workbench. The big advantage of this tool is that the workbench for the DSM is developed graphically. Therefore, knowledge about software development with Java, the graphical editor framework (GEF) or the graphical modeling framework (GMF) is not needed.

According to Hudak [7], programs written in a DSL are more concise, can be written more quickly, are easier to maintain and reason about. In the authors opinion, this list of advantages is also valid for domain-specific modeling. Furthermore, Hudak determines the basic steps for developing a own domain-specific language as

- Definition of the domain
- Design of the DSL capturing the domain semantics
- Provide support through software tools
- Create use-cases for the new DSL infrastructure

The approach described in this paper is presented according to theses steps in Section 3 and 4.

3 Approach

In this section, the domain specific modeling methodology for automotive mechatronics-based system development, with a focus on the open toolbox strategy, is presented. As mentioned in Section 2, details on the definition of the domain specific modeling can be found in [16]. Therefore, just a brief description is given in the following subsection.

3.1 Domain-Specific Modeling Language

The established SysML-based design method from [10] is extended by the newly developed *Embedded Mechatronics System Domain-Specific Modeling (EMS-DSM)* for the automotive embedded system design. The main goal of this methodology is to provide a lean approach for engineers to facilitate an embedded automotive mechatronics system modeling on a high abstraction level. The focus of the approach is on the model-based structural description of the E/E-System under development. Additionally, the signals and interfaces are an essential part of the modeling.

[3] https://eclipse.org/sirius/

The definition of the newly developed model-based domain specific language is shown in Figure 1. The top node *EMS-DSM Component* is the origin of all other classes at the language definition. Therefore, each of the derived classes inherits the five properties (*ID*, *Name*, *Requirement*, *Verification Criteria*, and *Specification*) from the base class.

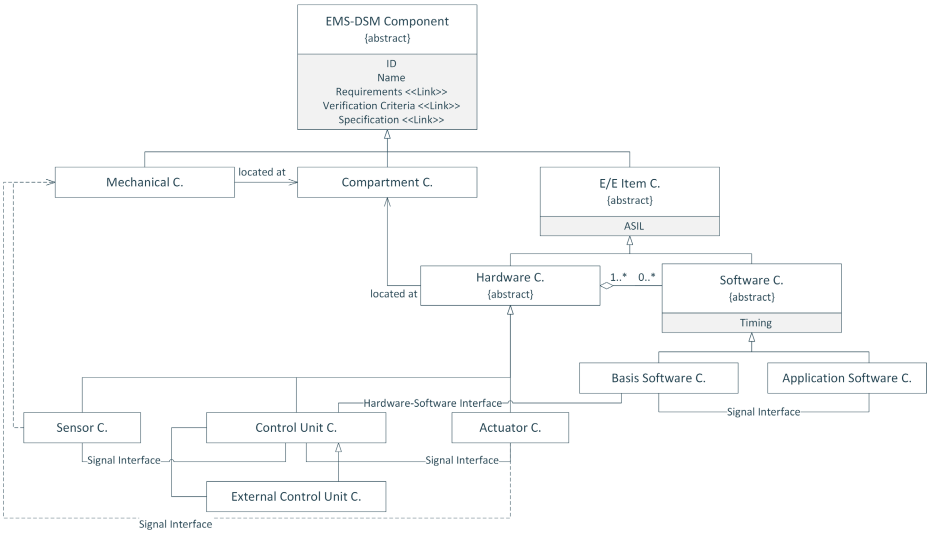

Fig. 1. *EMS-DSM* Definition (UML)

The language definition in Figure 1 represents the meta-domain of the model-based language. Subsequently, the EMS-DSM is tailored to the needs of the domain at the particular project or company. That is, design elements of possible types *Mechnical*, *Compartment*, *Sensor*, *Control Unit*, *Actuator*, *External Control Unit*, *Basis Software*, and *Application Software* are specified for the particular field of application. E.g. the domain of the presented application in Section 4 is *Embedded Mechatronics E/E-System Design for Compressed Natural Gas (CNG) Fuel Tank Systems*.

The EMS-DSM can be supported by a various number of tools, but at the time when the research project was initiated, a highest possible flexibility, as well as full access to the tools source code was desired. To achieve this, an own model editor (***E**mbedded **A**utomotive **Sy**stem Design*) has been developed, based on the open source project *WPF Diagram Designer* [5].

3.2 Traceability Between the Design and Other Artifacts

To achieve a lean development environment for automotive E/E-Systems, the whole engineering life cycle has to be supported. Therefore, not only the system architectural design, but also other artifacts, like requirements and test

case specification, are in the scope of this work. For topics like project management and requirements management, the web-based open source application *Redmine*[4] is used in this project. The de facto standard Automotive SPICE [2] defines three different types of requirements at the engineering process group: *Customer Requirements*, *System Requirements*, and *Software Requirements*. Out of the embedded E/E-System view, at least the hardware focus is missing. Additionally, requirements and design items regarding the mechanical components, have been introduced for the design of an embedded mechatronics-based E/E-System. Similar to the Automotive SPICE methodology on system and software level, engineering processes has been defined for these missing artifacts.

Section 3.1 contains the description of how the different types of designs (system level, software level, etc.) are created corresponding to the novel domain specific modeling. To achieve full traceability, these designs, respectively the various components at the designs, have to be linked to the corresponding requirements. This is accomplished by the *Requirements Linker* at EASy Design, which establishes a connection to the MySQL database, and therefore has full access to the requirements data at Redmine. More details about the requirements management capability of the presented project can be found at [15].

3.3 Open Toolbox Approach at the DSM

The main objective of the open toolbox strategy is to provide a possibility for the user to tailor the modeling item set to their particular needs. Every non-abstract EMS-DSM class from Figure 1 can be instantiated and utilized as type for a new toolbox item. By selecting one of the provided types, the behaviour of the new toolbox item is defined. E.g. if a new *Application Software Component* is created at the toolbox, the aggregation "*1..* 0..**" between *Hardware* and *Software Components* guarantees that the item can be used at the model within a *Hardware Component* only. These constraints contributes to the easy and intuitive handling of the modeling language. As mentioned previously, the defined modeling language and all presented features can be implemented on various modeling framework platforms, but at this project a self-made C# implementation has been preferred to achieve a highest possible grade of independence from third-party platforms.

To support the open toolbox methodology, additional functionality has been added to the custom-made software tool *EASy Design*. By selecting the command *Open Library Editor* at the menu bar, a new window (see Figure 2 in Section 4) is opened, offering toolbox modification options. At the window area *Create New Toolbox Item* the properties of the new toolbox item can be set. The drop down menu *Type* provides all non-abstract classes from the language definition. *Name* is a freely selectable identifier for the item, and *Mask* prompts the user to enter the path of a Portable Network Graphic (PNG), which determines the graphical representation of the toolbox item and its later appearance at the model. At the

[4] http://www.redmine.org/

window area *Delete Existing Toolbox Item* the no longer required item can be removed by choosing the respective name.

All library items are stored in an Extensible Markup Language (XML) file, corresponding to the following structure:

```
<EASyDesignLib>
  <LibItem>
    <Type></Type>
    <Name></Name>
    <Mask></Mask>
  </LibItem>
</EASyDesignLib>
```

4 Application

In this section the EMS-DSM approach with an open toolbox strategy is applied to the development of an automotive fuel tank system for compressed natural gas (CNG). For an appropriate scale of the use-case, only a small part of the real-world system is utilized. The application should be recognized as an illustrative material, reduced for internal training purpose for students. Therefore, the disclosed and commercially non-sensitivity use-case is not intended to be exhaustive or representing leading-edge technology.

To model the CNG fuel tank system, several mechanical, hardware, and software components are needed. As the main mechanical components, the following items being assumed to exist in the EMS-DSM library: *Tank Cylinder*, *Mechanical Pressure Regulator*, *Filter*, *Engine Rail*, and some *Tubing*. Moreover, four hardware components have already been added to the library: *In-Tank Temperature Sensor*, *CNG High Pressure Sensor*, *On-Tank Valve* (Actuator), and *Tank ECU* (Control Unit). So far, there are no software components at the library.

For a first draft of the system architectural design, an external control unit component *Engine ECU*, and a basis software component *CAN Driver* is needed. Therefore, the steps described in Subsection 3.3 are carried out for these new library items. The corresponding *Library Editor* windows are shown in Figure 2. The new library items are added at the EASy Design *Library Browser*, and the library file *EMS-DSM-Lib.xml* is extended by the following entries:

```
<LibItem>
  <Type>External Control Unit Component</Type>
  <Name>Engine ECU</Name>
  <Mask>"..\images\EASyLibExtEngECU.png"</Mask>
</LibItem> <LibItem>
  <Type>Basis Software Component</Type>
  <Name>CAN Driver</Name>
  <Mask>"..\images\EASyLibCANDriver.png"</Mask>
</LibItem>
```

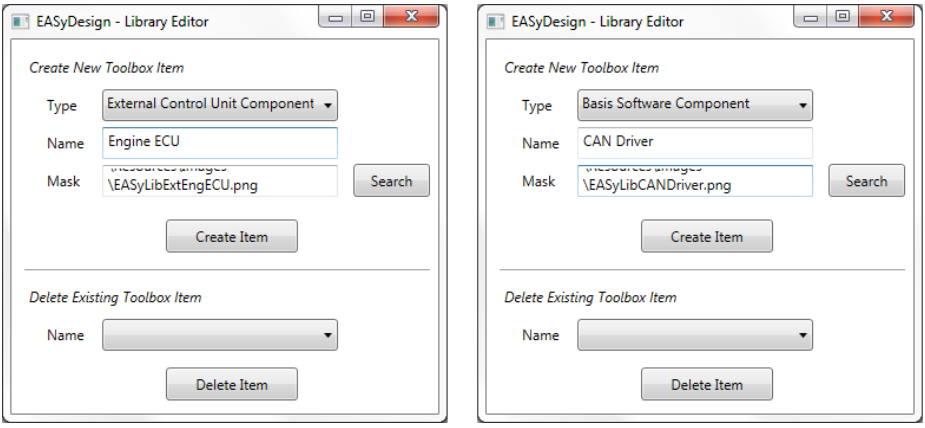

Fig. 2. Library Editor Windows for New Modeling Items

By adding these model items to the library, the system architectural design of the presented use-case can be created as shown in Figure 3. The CNG fuel tank system consists of seven mechanical components, which are blue coloured (Tank Cylinder, Filter, etc.) The medium flow between mechanical components, which is CNG in this use case, is displayed by blue lines with an arrow at the end. Furthermore, five hardware components are placed at the *System Architectural Design Model* level, which are yellow coloured (In-Tank Temperature Sensor,

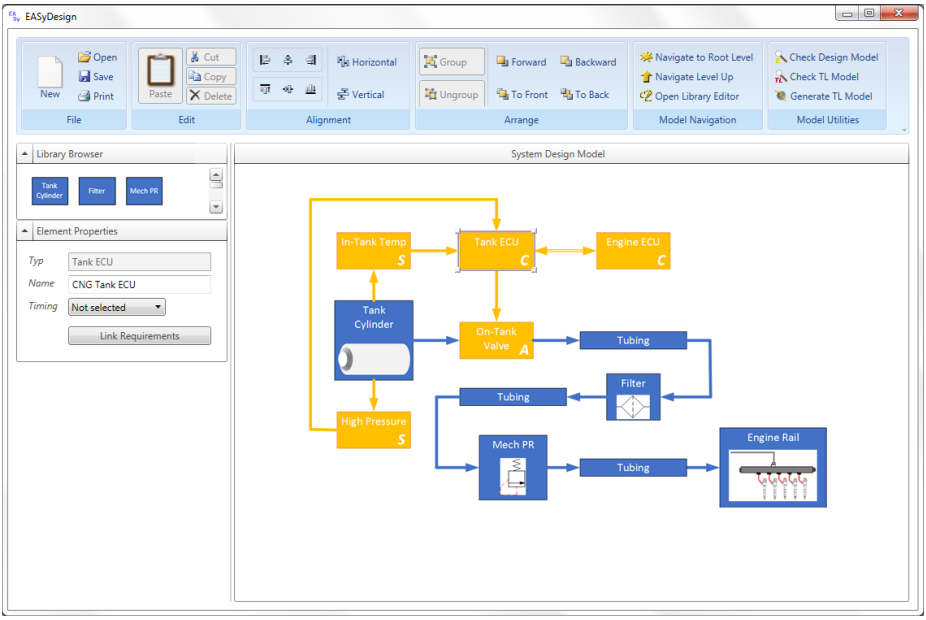

Fig. 3. CNG Tank System Architectural Design

Tank ECU, etc.) The signal flow between the components is displayed by yellow lines, ending with an arrow. Between the Control Unit and the External Control Unit component, a communication bus is inserted, characterized by the double compound line type and arrows on both ends.

As previously mentioned, the EMS-DSM definition requires at least one hardware component at the model to implement a software component. In this use-case the created basis software component *CAN Driver* shall be integrated at the *CNG Tank ECU*. With a double-click on the hardware component, the next modeling level is opened (named *E/E Item Design Level*), and the *CAN Driver* can be put in place.

5 Conclusions

In the previous sections, a lean method for the design of embedded automotive mechatronics-based E/E-Systems, with a focus on the open toolbox strategy, was presented. This approach has the potential to bring together the different engineering disciplines along the E/E-System development. Many artifacts like requirements, verification criteria, and various specifications can be linked to the models, created with the novel domain-specific modeling language. Supported by the linking of the artifacts, the vital traceability can be established. Depending on the respective tool chain and the organizations process landscape, the EMS-DSM models can also facilitate a single point of truth strategy.

By the model-to-model transformation mentioned in Section 2, a decision between the established SysML design techniques and the presented approach is not necessary. Instead, the EMS-DSM methodology can be utilized as an extension for mechatronics-based system designs to the existing tool chain. However, the possibility of modeling not only the system level, but also the software architectural level enables the presented work to be a standalone solution as well.

First use case implementations show promising results. However, there are several features on the open issue list, which have to be implemented in a next step. On the one hand, the options for describing the systems behavior, like e.g. some kind of task scheduling definition, shall be introduced. On the other hand, an advanced methodology for managing, as well as importing/exporting the signal interfaces has to be developed.

References

1. Andrianarison, E., Piques, J.-D.: SysML for embedded automotive Systems: a practical approach. In: Conference on Embedded Real Time Software and Systems. IEEE (2010)
2. Automotive SIG. Automotive SPICEProcess Assessment Model. Technical report, The SPICE User Group, Version 2.5 (May 2010)
3. Boldt, R.: Modeling AUTOSAR systems with a UML/SysML profile. Technical report, IBM Software Group (July 2009)

4. Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S., Ratiu, D.: Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments. Proceedings of the IEEE **98**(4), 526–545 (2010)
5. Code Project. WPF Diagram Designer - Part 4. Online Resource (March 2008). http://www.codeproject.com/Articles/24681/WPF-Diagram-Designer-Part (accessed March 2015)
6. Giese, H., Hildebrandt, S., Neumann, S.: Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. In: Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B. (eds.) Graph Transformations and Model-Driven Engineering. LNCS, vol. 5765, pp. 555–579. Springer, Heidelberg (2010)
7. Hudak, P.: Domain-specific languages. Handbook of Programming Languages **3**, 39–60 (1997)
8. ISO 26262, Road vehicles - Functional safety. International standard, International Organization for Standardization, Geneva, CH (November 2011)
9. Kawahara, R., Nakamura, H., Dotan, D., Kirshin, A., Sakairi, T., Hirose, S., Ono, K., Ishikawa, H.: Verification of embedded system's specification using collaborative simulation of SysML and simulink models. In International Conference on Model Based Systems Engineering (MBSE 2009), pp. 21–28. IEEE (2009)
10. Macher, G., Armengaud, E., Kreiner, C.: Bridging Automotive Systems, Safety and Software Engineering by a Seamless Tool Chain. In: 7th European Congress Embedded Real Time Software and Systems Proceedings, pp. 256–263 (2014)
11. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys (CSUR) **37**(4), 316–344 (2005)
12. Meseguer, J.: Why Formal Modeling Language Semantics Matters. In: Dingel, J., Schulte, W., Ramos, I., Abrahao, S., Insfran, E. (eds.) International Conference on Model-Driven Engineering Languages and Systems, MODELS 2014, Valencia, Spain. LNCS. Springer International Publishing Switzerland (2014)
13. Meyer, J.: Eine durchgängige modellbasierte Entwicklungsmethodik für die automobile Steuergeräteentwicklung unter Einbeziehung des AUTOSAR Standards. PhD thesis, Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik, Paderborn, Germany (July 2014)
14. Preschern, C., Kajtazovic, N., Kreiner, C.: Efficient development and reuse of domain-specific languages for automation systems. International Journal of Metadata, Semantics and Ontologies **9**(3), 215–226 (2014)
15. Sporer, H., Macher, G., Kreiner, C., Brenner, E.: A Lean Automotive E/E-System Design Approach with Integrated Requirements Management Capability. In: 9th European Conference on Software Architecture (ECSA 2015), Dubrovnik/Cavtat, Croatia (in press, 2015)
16. Sporer, H., Macher, G., Kreiner, C., Brenner, E.: A Model-Based Domain-Specific Language Approach for the Automotive E/E-System Design. In: International Conference on Research in Adaptive and Convergent Systems, RACS 2015, Prague, Czech Republic (2015) (under review)
17. Vujović, V., Maksimović, M., Perišić, B.: Sirius: A rapid development of DSM graphical editor. In: 18th International Conference on Intelligent Engineering Systems (INES), pp. 233–238. IEEE (2014)