COBEM 2021 Florianópolis - Brasil
ABCM Associação Brasileira de Engenharia e Ciências Mecânicas
26th ABCM International Congress of Mechanical Engineering
November 22-26, 2021. Florianópolis, SC, Brazil

# COB-2021-0222
# Goal-Oriented Requirements Analysis Applied to Systems Engineering in Automotive ECU Development

**Guilherme Bertelli**
Escola Politécnica, Universidade de São Paulo
Rua Prof. Melo Morais, 2231, São Paulo, Brazil
gbertelli@usp.br
Visteon Corporation
Rua Orlanda Bergamo 1062, Guarulhos, Brazil
guilherme.bertelli@visteon.com
**José Reinaldo Silva**
Escola Politécnica, Universidade de São Paulo
Rua Prof. Melo Morais, 2231, São Paulo, Brazil
reinaldo@usp.br

***Abstract.*** *Integrating Systems Engineering into complex systems development in an iterative and incremental software delivery approach is challenging given its latter's adaptive outlook. This challenge is further enhanced when the target system-to-be is an embedded system for the automotive industry, implying in other parallel activities in Mechanical Engineering, Hardware Engineering, Manufacturing, and DevOps, all directly serving as interfaces and constraints for the development. Requirements engineering value comes in the thoughtful analysis of stakeholder needs and technical specifications to define the system requirements, usually considering a defined scope and timing for system specification. In a Scrum-guided Software development approach, stakeholders change their vision, and so do system requirements and the product backlog. Therefore, requirements analysis tasks must be conducted iteratively throughout the system development, evaluating the changes and impacts on the system requirements, system architecture, manufacturing, mechanical, and hardware assessments. Hence, this paper proposes integrating a goal-oriented requirements analysis cycle in the Systems Engineering V Model and evaluating the impacts across the process phases (system architecture, system requirements, development, testing) in the context of automotive ECU (Electronic Control Unit) development. The Goal-Oriented Requirements Engineering relies on semi-formal modeling using KAOS, which is used as a method for requirements specification organization, representing high-level goals interdependence. The result is a framework that integrates goal modelling into Systems Engineering V-Model Concept in an Agile software development project. The case study brings some insights into using goals as an important factor for traceability - which is crucial to maintenance, and how goal-oriented requirements analysis can aid in adding value do the product increment.*
***Keywords:*** *Goal-Oriented Requirements Engineering, Systems Engineering, V Model, KAOS, Embedded Systems, Automotive ECU, Scrum*

## 1. INTRODUCTION

Latest regulations and trends in the automotive industry are shaping the future of the market, with significant investments being placed into electric vehicles, autonomous driving, and connected car features (Pacheco, 2021). In that context, the level of innovation expected on the systems development is higher, and with innovation comes the challenges of unknown needs, incomplete requirements, and delayed clarifications. Considering the already complex environment of automotive ECU development (Broy, 2006), with thousands of system integration, vehicle diagnostics, and non-functional requirements (functional safety, cybersecurity, environmental, and similar issues), the adaptive factor of the development process must be well considered.

Within the Systems Engineering approach, and considering an iterative delivery-based embedded systems development, the sprint-based workflow calls for an evolving, customer-oriented product backlog. Therefore, the expectation is that requirements be updated based on innovative features, newly acquired knowledge, and new business needs. That requires impact analysis on all sub-systems and interfaces, manufacturing provisioning, system testing tools, and program timing and resources (Pretschner *et al.*, 2007). In that direction, considering the evolution of requirements during the initial development phase should demand other requirements analysis cycles up to the implementation (dynamic requirements). With intensive competition, rapid technological advances, and fluid market demands, agility is imperative (Bianchi *et al.*, 2020).

This paper is motivated by this need and proposes a life cycle analysis and framework integration in the systems engineering V Model. The Requirements Engineering analysis is goal-oriented (using KAOS semi-formal modeling), allowing for a subsequent formalization, followed by a review. These steps must be conducted within a systems architecture-driven view. The goal is to provide a process that allows for innovation and requirements evolution during the development phase while ensuring a concise impact evaluation on all affected sub-systems and interfaces impacting all systems engineering V Model phases.

This paper will be presented as follows: Section 2 will bring the theoretical background in systems engineering applied in iterative-based software development while also noting fundamental aspects of model-based requirements engineering and goal-oriented requirements engineering; Section 3 will explore the relevant related works on frameworks and processes proposals applied to systems engineering, mostly on innovation-heavy software-based projects; Section 4 will describe the proposal in detail, from the semi-formal modeling up to review, and the integration in the V model within the Scrum framework, using a practical case-study of an Infotainment system development; Section 5 concludes with final considerations and future work.

## 2. SYSTEMS ENGINEERING IN AUTOMOTIVE EMBEDDED SOFTWARE DEVELOPMENT

In the last twenty years, embedded software systems have become the primary effort of automotive product development, enabling advanced driver assistance, connected services features, and applying software solutions to previously hardware-based ones. With software driving innovation, together comes the rapid evolution of requirements and complexity of the system. Such tendency, alongside the increasing speed of time-to-market and uncertainty in markets, creates a challenging environment for the Systems Engineering processes (Stelzmann *et al.*, 2010).

The Systems Engineering V Model is a sequential method used to visualize key SE areas (INCOSE, 2015). It is applied considering a fixed scope of requirements defined during the conceptualization - which are then analyzed, refined, and taken to the following phases: high-level design (software architecture), detailed design, implementation, integration and testing, subsystem verification (unit/component testing), system verification, operations and maintenance (Elm *et al.*, 2008). The V Model schema can be seen in Figure 1. In principle, it means that new needs, technical findings on interoperability, or additional use cases would result in demands for a formal change request, with subsequent impact analysis and commercial discussion, colliding with the Scrum framework pillars.

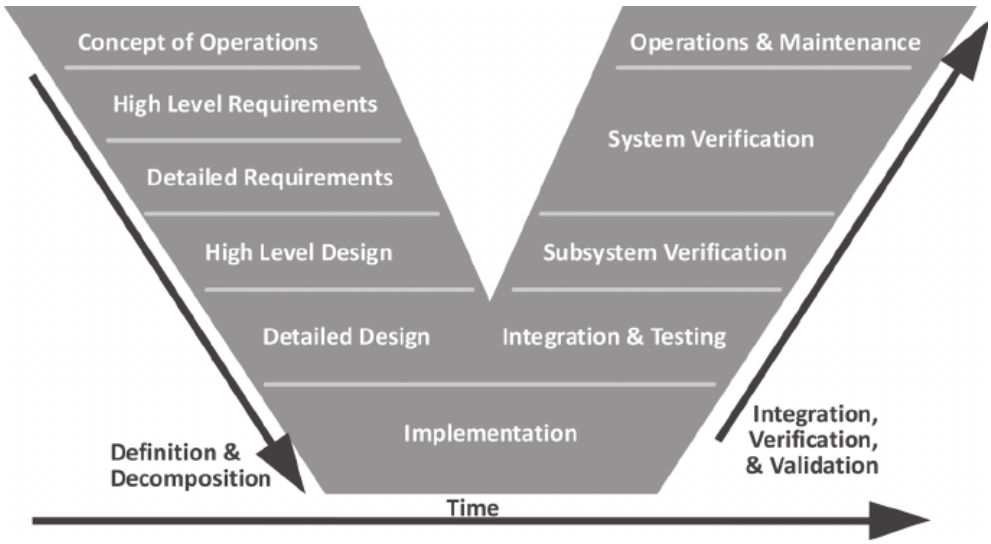


Figure 1. Systems Engineering V Model (Elm *et al.*, 2008)

The Scrum framework was designed for agile software development, mainly deriving from extreme programming, and employs an iterative, incremental approach to optimize predictability and control risks (Schwaber and Sutherland, 2020). With its pillars being transparency, inspection, and adaption, the framework focus on people (and stakeholder needs) rather than documentation. This agile approach delivery is based on sprints (typically one month) and specific events (sprint planning, sprint review, sprint retrospective, and daily scrum) to add value to the product on every delivery, considering a constantly evolving product backlog (Schwaber and Sutherland, 2020).

In Systems Engineering-guided development, the product backlog consists of work packages, or system features, split throughout the sprints for implementation, for which system requirements need to be defined and documented. Therefore, if an evolving backlog is considered, the requirements shall be constantly revisited, updated, and refined to add value to the product and/or include a neglected use case. In other words, when changes appear, the process should provide routines to adapt the product (Stelzmann *et al.*, 2010). On a macro view, this would imply having the V-model being applied to

every sprint, as suggested in (Takahira *et al.*, 2014).

Automotive ECUs are highly complex systems, consisting of several different hardware components, interfacing with embedded software, implementing multiple interdependent features Pretschner *et al.* (2007). These internal functions communicate with external systems over different communication links ruled by different topologies (CAN network, Ethernet networks, MOST networks, and others) and protocols (I2C, CAN, TCP, UDP, HTTP, MQTT), which are also interdependent. Additionally, the automotive systems are ruled by many different non-functional requirements, from strict performance specifications by the OEM to country-specific regulations and domain-relevant standards (functional safety, cybersecurity, etc.). Also, the same developed ECU is expected to work in different vehicles, with different architectures and partner-ECUs interoperability requirements add another layer of software complexity and diagnostics configuration. Given this complex system context, the systems engineering activities need a sound requirements analysis cycle during product development to guarantee a precise impact analysis of requirement updates on all system levels.

For enabling agile systems engineering for the development of a complex system, Model-Based Systems Engineering (MBSE) can serve as a surrogate for rapid change and continuous verification on an agile framework (Bott and Mesmer, 2020). The system model provides the needed agile architecture re-design and continuous integration and testing - characteristics from MBSE which fit in Scrum Montgomery (2013). In this direction, Model-Based Requirements Engineering (MBRE) shall be the core approach for the system engineering activities, as it can help overcome some of the difficulties related to natural language requirements (Pretschner *et al.*, 2007) and provide means for automation and consistency checks.

### 2.1 Model-Based Requirements Engineering

Model-based Requirements Engineering (Silva *et al.*, 2021) consists of a model-based iterative requirement analysis process in place of the usual text-based requirements. Model-based requirements are more formal, consistent, and visual, while also more susceptible to checking, simulation, and verification (Mordecai and Dori, 2017), which makes it a consistent fit for agile development (Bott and Mesmer, 2020). Effective solutions to software development demands may be achieved by combining requirements engineering processes and model-driven development (Soffer and Dori, 2013).

Requirements modeling procedure means designing an abstract, formal and unambiguous representation of the textual system requirement, which supports effective communication among the system development process fitting a cohesive concept. Subsequently, as requirements modeling is integrated into the workflow of the development process, it promotes collaboration between systems, validation, and engineering development (Soffer and Dori, 2013). In this paper, the surrogate approach to the requirements process is goal-oriented requirements engineering (GORE), where the requirements modeling is based on the definition of system goals.

#### 2.1.1 Goal-Oriented Requirements Engineering

A goal is an objective the system-to-be has to achieve (van Lamsweerde, 2001). Therefore, the Goal-Oriented Requirements Engineering is based on the identification of system goals and the transformation of these goals into requirements (Horkoff *et al.*, 2016). Using goals in requirements engineering brings significant benefits, such as a precise criterion for completeness and pertinence, efficient traceability from strategic business needs to technical requirements and a natural process for managing complex technical documentation (van Lamsweerde, 2001).

This paper will be using the Knowledge Acquisition in Automated to Specification (KAOS) to represent Goal modeling diagrammatically. The KAOS model provides semi-formal diagrams to model goals, requirements, operations, and agents, semantically linked. The system's goals are elicited through incremental "why's" and "how's" questions and captured in a hierarchical diagram using and/or decomposition of high-level goals into detailed requirements (Zickert and Beck, 2010).

### 2.2 Related works

Among the state-of-the-art publications proposing design processes or frameworks for requirements analysis in automotive systems development, some of the most relevant studies will be highlighted in this section. First, Inkermann *et al.* (2019) proposed an MBSE concept based on five partial models representing emerging automotive systems from abstract concept to systems architecture. These partial models are use cases, functions, function realization, system structure, and product structure. The model relied on UML and SysML, and the case study was based on an electric vehicle.

In Takahira *et al.* (2014), a framework was defined for integrating Scrum into the systems engineering V Model. The framework consists primarily of three phases: the first phase is the product vision definition by the Product Owner and customer requirements; the second step is the project planning and team definition for all V model activities; and the third step is the Scrum Sprint, which goes through the entire V cycle phases and generates product delivery. The steps are then repeated for each sprint. However, the requirements engineering approach was not specified, and the case study was a safety-critical automotive ECU development.

Holder *et al.* (2017) work presented a process to integrate MBSE in product development using graph-based design

languages. The framework defines four steps: Customer needs, requirements, geometry synthesis, and simulation. The requirements step utilizes class diagrams, which are formalized with a graph-based representation. The developed system used as an example is a gear system for Electrical Multiple Units (EMU) applied to trams, subways, or high-speed railways.

Morandini *et al.* (2015) presented a requirements engineering framework for adaptive software systems, considering goal modeling as a reference. The approach combines goal-oriented concepts with high variability design methods - and makes use of a proprietary defined modeling language, which borrows concepts from *i**. The goal modeling is taken in 4 steps: environmental modeling, conditions modeling, detailed goal modeling, and refined conditions modeling. The requirements modeling case study is a controller for a cleaning robot.

On past papers by Postigo and Silva (2020) and Silva *et al.* (2019), KAOS modelling was also used. The KAOS model was first used to model a microgrid system implemented in the Amazon forest in the forming work. The sub-goals were refined to the requirements or expectations, allowing the identification of missing emergent requirements generated in a system engineering approach. The latter work uses KAOS to model the car sequencing problem - that is, to control the daily production of car manufacturing. The goal model was formalized in Petri Net, aiming for a more sophisticated formal verification of the manufacturing system, applicable to service-oriented digital manufacturing. Further research by Silva *et al.* (2021) presented an MBRE requirements life cycle, pictured in Figure 2. The proposal conducts a semi-formal goal-modeling on elicitated requirements, which are formalized in Petri Nets - aiming for verification and convergence. The Petri Net formalization is then reviewed for consistency and closeness.

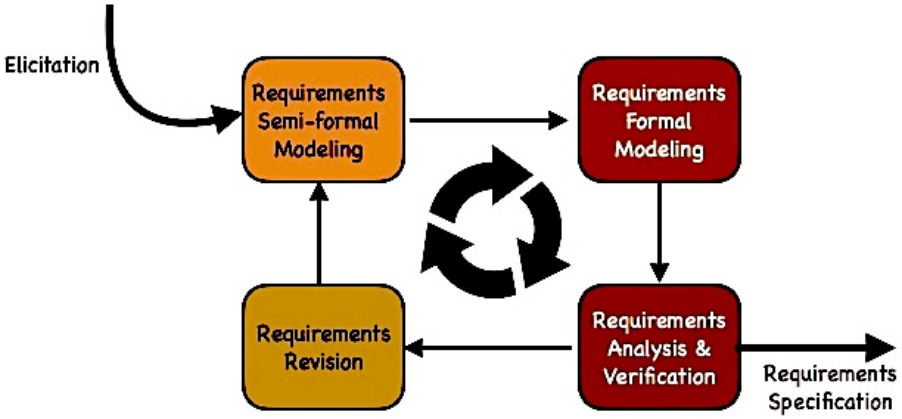


Figure 2. MBRE process cycle (Silva *et al.*, 2021).

The proposal on this paper differs from the above mentioned by conducting goal-oriented semi-formal modeling in KAOS, followed by integration in a Scrum-oriented V Model product development. The focus is on the requirements analysis cycle, which is an earlier stage of the system specification. Moreover, the framework is intended as a solution for systems engineering development, evaluating the impact across V Model phases.

## 3. GOAL-ORIENTED REQUIREMENTS ANALYSIS FRAMEWORK

This section will present our proposal, using a practical case study as a guide. Once the project concept is defined and the Systems Engineering team has enough inputs (documentation, technical workshops), the system shall be modeled in a goal-oriented KAOS diagram. Hence, Figure 3 exhibits the KAOS model of an Infotainment system, with a certain level of abstraction considered given the scope of this paper. High-level goals are split into sub-goals.

The main goal of a central infotainment system is providing information and entertainment to vehicle users. This service can be split into four sub-goals required to fulfill the main goal: Play audio, provide a visual interface, integrate with the vehicle, and integrate with manufacturing. The arrow direction states the goal hierarchy, and multiple sub-goals combined into a higher goal instantiate an AND condition. In order to play audio, the system needs to support project-specific media requirements, and for this example, these are Bluetooth, USB, and Tuner (AM/FM). Integration with vehicles demands a physical CAN network interface and physical analog audio outputs. The visual interface provisions shall minimally provide the user with information about the currently playing media information and allow the user to use change/play/pause functions. Furthermore, important vehicle information should be present on the HMI (for example, consumption, tire monitoring, proximity sensors). A necessary goal for an automotive ECU is to integrate with manufacturing, that is, be configurable according to specified commands and provide diagnostics feedback for part health monitoring. Figure 4 shows Visteon's InfoCore Infotainment system.

In Figure 3, the darker boxes stand for "soft goals" or non-functional requirements. For instance, for all standardized technologies, it is expected for it to comply with the standard. An infotainment system is a telecommunication system that should comply with the regulatory requirements and vehicle safety thread analysis and risk assessments. Lastly, the

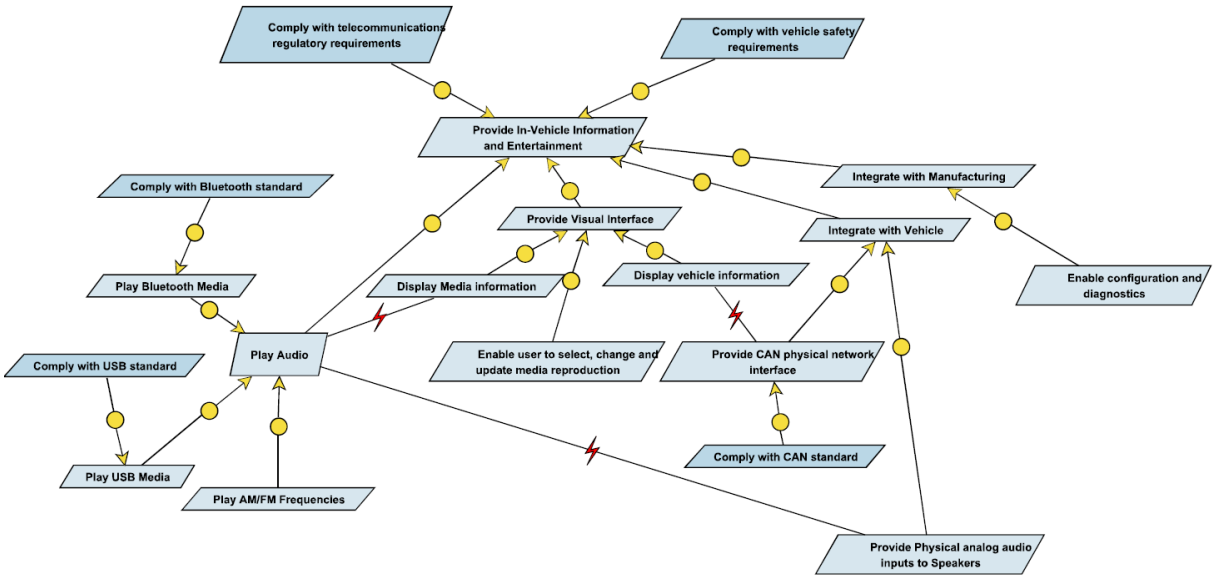


Figure 3. Infotainment System Simplified Goal Model

red lightning connections indicate conflicts. That is, the goals are interdependent and compete for the same resources. For instance, the system will only be able to display media information if it supports audio reproduction. Audio can only be played in the vehicle speakers if an appropriate audio connection is provided. Vehicle information can only be displayed if this information is received from the vehicle over a CAN network bus. It is important to note that this is a simplified goal model just for exemplification. Many other sets of sub-goals were abstracted for the scope of this paper, such as additional common features, like a remote user interface (phone mirroring) and connectivity with cloud/proprietary back-ends. Likewise, sub-goals could be further split, and further goal conflicts could be identified (for example, end-of-line commands typically use UDS over CAN).


Figure 4. Visteon's Infocore Infotainment System. Source: https://www.visteon.com/products/infotainment/ Accessed 6 June 2021.

Goals provide a solid, high-level orientation for systems design. Goal-modelling also provides a straightforward visualization of dependencies with other goals. Therefore, the proposal uses the initial goal-modeling to organize the requirements documentation in a requirements management tool, with the sub-goals representing the Requirements Specification to baseline complying documents. These documented complying requirements are expected restrictions to implement the system goals and compose the backlog items on each Scrum Sprint. The Sprint backlog will serve as input artifacts to be taken for a high-level design (software architecture), detailed software module design, and implementation. Figure 5 pictures this framework.

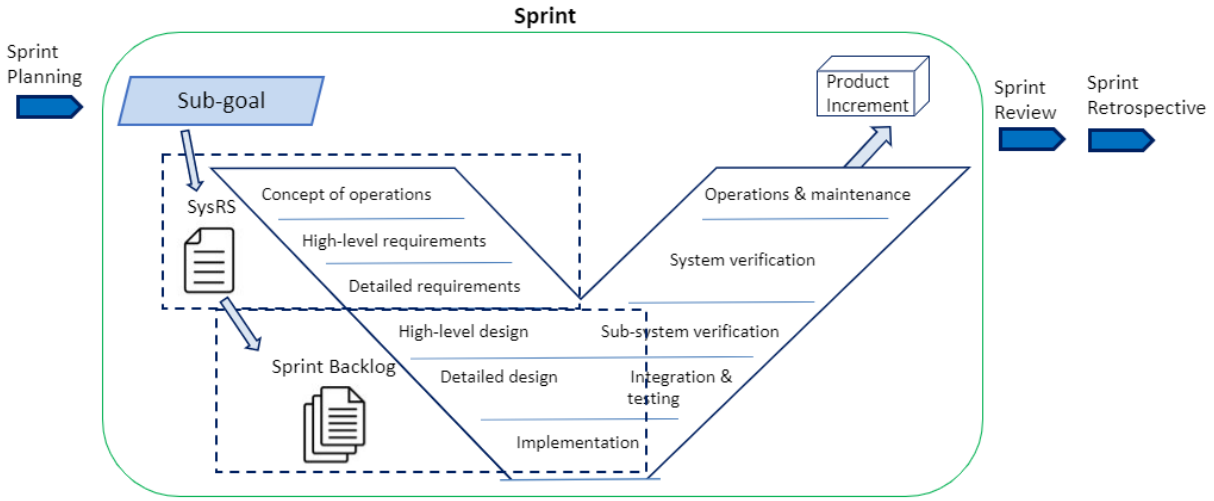


Figure 5. Goal to Spring backlog in the V Model

When dealing with goals as a driver on a goal model, the dependencies between sub-goals can be visually identified - That is, once capturing the requirements in a SysRS (System Requirements Specification), the interfaces between sub-systems will be defined and properly specified. Once in the concept of operations, this will ensure all the correct domain specialists will be in the technical workshops to define the use-cases and end-user impacts. Once the high-level requirements are identified and clarified, the previous step leads to delivering applicable documentation with the proper balance between functional and non-functional requirements. Non-functional requirements are derived from the main goals or are instantiated by the design environment's restrictions.

Moving on to detailed requirements, the capture of systems requirements on a goal-based SysRS denotes significant dependencies clash among sub-goals, which will aid during the definition of the artifact's attributes and associate the impacts updating existing requirements or adding new ones. These detailed requirements will be part of the features delivered through the spring backlog - first from a software architecture standpoint, moving to detailed software component design and implementation. The major benefit here is bi-directional traceability between a software function to system goal (or sub-goal), going through system architecture, system specification, and software architecture. Carrying forward through integration and testing, sub-system verification, and system verification will all be raising defects and non-conformance towards an end-user feature traced to a system goal. The same applies to Operations and Maintenance, which will primarily demand product increment delivery to customers, bug-fix plan, manufacturing operations, training, reviews, audits, and similar features.

Providing well-defined bi-directional traceability towards a system goal is effective, as goals are a solid system characteristic. Although features, functions, and detailed requirements can change and impact (or even change) lower-level goals, this can still be linked to higher-level system goals. In addition, analyzing traceability to a product's high-level goal facilitates communication between different stakeholders of multiple areas (Aljahdali *et al.*, 2011), enabling efficient communication when discussing high-level technical aspects. Also, when elaborating on the product increment, which is the outcome of the sprint and customer deliverable, a clear set of goals should also assist on defining the features that mostly bring value to the product. Therefore, from a theoretical perspective, higher-level goals work as a direct representation of dynamic requirements. Those requirements are invariant and persist valid during the whole design process, directly impacting the management of the project and documentation.

The requirement cycle proposed can also admit formalization, which would be an excellent option to complex project demands. Typically, tools that support KAOS and GORE methods are formalized using LTL (Linear Time Logic). Recent works transferred KAOS diagrams to Petri Nets (Silva *et al.*, 2021), which could fit better the demand on the automotive industry. Indeed, it would be advantageous to verify the requirements in each cycle - and when emergent requirements and use cases appear to be included.

## 4. CONCLUSIONS

This paper presented a goal-oriented requirements engineering proposal applied to a system engineering context considering a Scrum-guided embedded software development of Automotive ECUs. The article introduced the challenges and complexity of automotive ECU development in an innovative scenario and related works that elaborated on model-based solutions to engineering problems. This research suggested that using goal-oriented models at the early-phase requirement engineering method can support the system specification execution when dealing with evolving requirements, providing

efficient bi-directional traceability through systems engineering phases. Furthermore, the goal orientation can identify the key features that should consist of a product increment, adding value to the product increment.

Practical use in industrial software teams would depend on software tools to support requirements modeling in KAOS and help in an eventual need for formalization and verification. Such tools already exist. It would also be necessary to support documentation coupled with the V-model or with a cascade of "mini V's." Such a tool capable of automating documentation and negotiation of improvements does not exist yet. It would be based on the communication with the other tools using XML representations for KAOS and Petri Nets.

For future works, it is intended to formalize the semi-formal KAOS model semantically transferring the diagrams to Petri Nets and expand the requirements engineering analysis impacts in more detail throughout systems engineering phases and product development life cycle. Such development will be integrated into a more elaborated framework considering all steps presented in Figure 2. The challenge is to design and implement the support system for documentation and negotiation based on this formalization.

## 5. REFERENCES

Aljahdali, S., Bano, J. and Hundewale, N., 2011. "Goal oriented requirements engineering -a review". *Proceedings of the ISCA 24th International Conference on Computer Applications in Industry and Engineering, CAINE 2011*.

Bianchi, M., Marzi, G. and Guerini, M., 2020. "Agile, stage-gate and their combination: Exploring how they relate to performance in software development". *Journal of Business Research*, Vol. 110, pp. 538–553.

Bott, M. and Mesmer, B., 2020. "An analysis of theories supporting agile scrum and the use of scrum in systems engineering". *Engineering Management Journal*, Vol. 32, No. 2, pp. 76–85.

Broy, M., 2006. "Challenges in automotive software engineering". In *Proceedings of the 28th International Conference on Software Engineering*. Association for Computing Machinery, ICSE '06, p. 33–42.

Elm, W.C., Gualtieri, J.W., McKenna, B.P., Tittle, J.S., Peffer, J.E., Szymczak, S.S. and Grossman, J.B., 2008. "Integrating cognitive systems engineering throughout the systems engineering process". *Journal of Cognitive Engineering and Decision Making*, Vol. 2, No. 3, pp. 249–273.

Holder, K., Zech, A., Ramsaier, M., Stetter, R., Niedermeier, H.P., Rudolph, S. and Till, M., 2017. "Model-based requirements management in gear systems design based on graph-based design languages". *Applied Sciences*, Vol. 7, No. 11.

Horkoff, J., Aydemir, F.B., Cardoso, E., Li, T., Maté, A., Paja, E., Salnitri, M., Mylopoulos, J. and Giorgini, P., 2016. "Goal-oriented requirements engineering: A systematic literature map". In *2016 IEEE 24th International Requirements Engineering Conference (RE)*. pp. 106–115.

INCOSE, 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Vol. 4. John Wiley and Sons, Inc, Hoboken, NJ, USA.

Inkermann, D., Huth, T., Vietor, T., Grewe, A., Knieke, C. and Rausch, A., 2019. "Model-based requirement engineering to support development of complex systems". *Procedia CIRP*, Vol. 84, pp. 239–244.

Montgomery, P.R., 2013. "Model-based system integration (mbsi) – key attributes of mbse from the system integrator's perspective". *Procedia Computer Science*, Vol. 16, pp. 313–322. 2013 Conference on Systems Engineering Research.

Morandini, M., Perini, A., Penserini, L. and Marchetto, A., 2015. "Engineering requirements for adaptive systems". *Requirements Engineering*, Vol. 22.

Mordecai, Y. and Dori, D., 2017. "Model-based requirements engineering: Architecting for system requirements with stakeholders in mind". In *2017 IEEE International Systems Engineering Symposium (ISSE)*. pp. 1–8.

Pacheco, P., 2021. "What are the top five automotive trends for 2021?" Automotive World, https://www.automotiveworld.com/articles/what-are-the-top-five-automotive-trends-for-2021/ Accessed 31 March 2021.

Postigo, M.A.O. and Silva, J.R., 2020. "Microgrid system design based on model based systems engineering: the case study in the amazon region". In *Proc. of Brazilian Sumposium in Electric Systems*.

Pretschner, A., Broy, M., Kruger, I.H. and Stauner, T., 2007. "Software engineering for automotive systems: A roadmap". In *Future of Software Engineering*. pp. 55–71.

Schwaber, K. and Sutherland, J., 2020. "The scrum guide - the definitive guide to scrum: The rules of the game". https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf zoom=100 Accessed 7 May 2021.

Silva, J.M., Javales, R. and Silva, J., 2019. "A new requirements engineering approach for manufacturing based on petri nets". *IFAC-PapersOnLine*, Vol. 52, pp. 97–102. doi:10.1016/j.ifacol.2019.10.006.

Silva, J.R., Macedo, E., Correa, Y. and Medeiros, R., 2021. "A multilayer proposal to a smart home applied to healthcare". *Polytechnca*, Vol. 4, pp. 1–14.

Soffer, A. and Dori, D., 2013. *Model-Based Requirements Engineering Framework for Systems Life-Cycle Support*, Springer Berlin Heidelberg, pp. 291–311.

Stelzmann, E., Kreiner, C., Spork, G., Messnarz, R. and König, F., 2010. "Agility meets systems engineering: A catalogue

of success factors from industry practice". *Systems, Software and Services Process Improvement. Communications in Computer and Information Science*, Vol. 99, pp. 245–256.

Takahira, R.Y., Laraia, L.R., Dias, F.A., Yu, A.S., Nascimento, P.T.S. and Camargo, A.S., 2014. "Scrum and embedded software development for the automotive industry". In *Proceedings of PICMET '14 Conference: Portland International Center for Management of Engineering and Technology; Infrastructure and Service Integration*. pp. 2664–2672.

van Lamsweerde, A., 2001. "Goal-oriented requirements engineering: a guided tour". In *Proceedings Fifth IEEE International Symposium on Requirements Engineering*. pp. 249–262.

Zickert, F. and Beck, R., 2010. "Evaluation of the goal-oriented requirements engineering method kaos". *16th Americas Conference on Information Systems 2010, AMCIS 2010*, Vol. 7, pp. 5443–5452.

## 6. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.