Development of an Agile Concept for MBSE for Future Digital Products through the Entire Life Cycle Management Called Munich Agile MBSE Concept (MAGIC)

Vahid Salehi

Institute of Engineering Design of Mechatronic Systems and PLM, Munich
Salehi-d@hm.edu

Abstract. Model-based systems engineering (MBSE) is a well-known approach and one of the new aspects for system development in gaining the control over the complexity of systems and the development processes, while agile is a project management methodology originally from software development that uses short development cycles to focus on continuous improvement in the development of a product or service. In this paper, the different kinds of V-Models have been analyzed and the missing aspects related to new approaches based on digitization such as integration of IOT parts of data from manufacturing or usage of the products or data form services have been elaborated. Moreover, this paper will present an approach which melds the new development processes like DevOps or agile product development to a new MBSE Framework. In this paper, we adopt the concept of agile into MBSE and then proposed the new approach - Munich Agile MBSE Concept (MAGIC). The highlights of MAGIC approach can be concluded as 1) the requirements are continuously adopted to every stage of system development from first product ideas up to manufacturing and product services/usages for a better traceability 2) an agile working and communication are implemented between every state of the MAGIC approach. This ensures to have a close connection during system development phases. 3) the idea of Industry 4.0 has been included and reflected to achieve automation and data exchange with manufacturing technologies. 4) the concept of IOT (Internet of Things) is also considered when it comes to the usage and service of the system to satisfy the customer's needs. 5) the whole spirit of agile is reflected as the iterative and incremental design and development. Last but not least, a use case of Mars rover development project has been introduced to verify the feasibility of this approach.

Keywords: Model-based systems engineering (MBSE), agile product design, Munich Agile MBSE Concept (MAGIC), Product lifecycle management (PLM)
DOI: https://doi.org/10.14733/cadaps.2020.147-166

1 INTRODUCTION

Model-based systems engineering (MBSE) is well-known in gaining control over the complexity of systems and the development processes [4][12]. The basic idea of MBSE is to formalize the system description as well as to connect the relevant information - needed for the creation of various artifacts during the system development - in a system model [18]. Furthermore, a standardized and formalized language like OMG Systems Modeling Language (OMG SysML) is necessary for the formal definition of the system [11]. However, traditional MBSE approach usually works professionally for the project on a relatively large scale yet is time-consuming. Additionally, the entire process chain with production and manufacturing stage as well as the usage and service stage are usually neglected. Moreover, as increasing implementation and integration have been applied to MBSE, it is noticeable that requirement which builds up the foundation of the whole project should be tracked and approved at the entire lifecycle of the development chain. Therefore, to pursue a new approach, in many cases, seems necessary.

Contemporary manufacturing companies have been confronting with various demands such as reducing time-to-market, flexibly changing operation schedule, which is hard to be satisfied concurrently [17]. Agile is a project management methodology that uses short development cycles called “sprints” to focus on continuous improvement in the development of a product or service [7]. Although agile methods were developed by and for software developers, the discourse on their relevance for other types of development projects has been intensifying the past years and receive increasing attention in the development of mechatronic products, however as the boundary conditions are different (e.g. virtual vs. physical prototypes) modified forms established themselves. Therefore, in this paper, we intended to adopt the concept of agile from software development and combined with MBSE approach into cyber-physic product design. A new approach - Munich Agile MBSE Concept (MAGIC) has been proposed.

MAGIC concerns about the requirement at each stage of the product development lifecycle. Moreover, besides the traditional development stages such as system functional design, system architecture design, and system verification and validation, production stage, product usage, and service stage is also been considered in the entire lifecycle. Different from typical V-model used widely in systems engineering, MAGIC has the shape of infinity loop to demonstrate that process chain of each stage does not end when the product has been delivered and applied by the customer. On the contrary, it will keep going back to the first requirement stage when the user has a new demand for the product.

The remainder of the paper has the following structure: Section 2 describes the current MBSE methods which support most of the current development projects as well as their challenges for the development of the cyber-physical system in the future. Which leads to a detailed description of the MAGIC approach in Section 3 to elaborate the structure and concepts. Section 4 contains a use case implementation of Mars rover based on the MAGIC approach proposed in Section 3, which reflects and approves the feasibility and value of the proposed approach. Finally, Section 5 presents the conclusions of this paper and future work.

2 STATE OF THE ART

2.1 Agile Product Development

To assure that evolving customer requirements can be met and costs from late changes can be avoided, companies increasingly develop products on the basis of agile product development processes. Agile processes are characterized by a sequence of iterative sprints, in which parallelized activities are undertaken. At the end of each sprint, an incremental prototype of the product or various subsystems is created so that during the entire process the degree of maturity of the product is continuously and successively increased [21]. Each incremental prototype increases the functionality or features or confidence of the previous prototype generation. This enables early evaluation of subsystems or components. Incremental development allows also for early feedback of customers. Agile development teams need to possess all the necessary skills to develop and validate the product. Popular agile approaches are briefly listed as a knowledge base for this goal:

- Design Thinking, which focusses on gaining a deepest possible understanding of the end-user's needs, and aims at a rapid conception of solutions, an implementation in prototypes and early testing for further knowledge gain [6].
- Human Centered Design, which works with the users to gain a holistic understanding of the design problems that are addressed to develop systems, which are useful, usable and desirable for humans with the consistent focus on higher levels of humans' needs **Error! Reference source not found.**.
- ASD - Agile Systems Design is a holistic, structured approach for the agile development of mechatronic systems, the associated product strategy, validation systems, and production systems, consisting of principles, methods, and processes of PGE - Product Generation Engineering. It focuses on the consequent integration of PGE into the development process and creates a situation-specific balance between structuring and agile elements [1]. Here we mainly introduce the 'Scrum' method and 'DevOps' approach.

2.2 Scrum

In general, Scrum is an agile framework for managing work with an emphasis on software development to support the improvement of project management and development practices to develop and sustain complex products. It is designed for teams of three to nine developers who break their work into actions that can be completed within timeboxed iterations, called sprints and track progress and re-plan in 15-minute stand-up meetings, called daily scrums. Approaches to coordinating the work of multiple scrum teams in larger organizations include Large-scale Scrum (LeSS), Scaled Agile Framework (SAFe), a scrum of scrums, and Scrum@Scale, among others [31]. It has been used by leading software companies for object-oriented software and in product engineering as an agile process of trial and error to develop new systems within a team of non-experts [32].

Scrum is a powerful framework for implementing agile processes in software development and other projects. This highly adopted framework utilizes short iterations of work, called sprints, and daily meetings, called scrums, to tackle discrete portions of a project in succession until the project as a whole is complete. Scrum relies on a self-organizing, cross-functional team. The scrum team is self-organizing in that there is no overall team leader who decides which person will do which task or how a problem will be solved. Those are issues that are decided by the team as a whole [21]. There are three key roles within Scrum:

The Scrum master, product owner, and Scrum team members:

- The product owner creates and prioritizes a product backlog.
- Teams select items from the backlog and determine how to complete the work.
- Work must be completed within a sprint.
- The Scrum master meets with teams briefly each day to get progress updates.
- Sprint reviews are conducted at the end of each sprint.
- The process starts again until all work or backlog is complete.

2.3 DevOps

DevOps is the combination of cultural philosophies, practices, and tools that increase an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market. Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function [3].

Transitioning to DevOps requires a change in culture and mindset. At its simplest, DevOps is about removing the barriers between two traditionally siloed teams, development, and operations. In some organizations, there may not even be separate development and operations teams; engineers may do both. With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations. They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers. They take full ownership for their services, often beyond where their stated roles or titles have traditionally been scoped by thinking about the end customer’s needs and how they can contribute to solving those needs. Quality assurance and security teams may also become tightly integrated with these teams. Organizations using a DevOps model, regardless of their organizational structure, have teams that view the entire development and infrastructure lifecycle as part of their responsibilities.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps. These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity.

2.4 Model-based Systems Engineering (MBSE) with V-model

INCOSE [14] defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical and software". MBSE is not only considering the different needs of different stakeholders [2] but also supporting the entire development process [35]. In general, V-model and their derived versions are commonly used as the method instruction for MBSE approach since V-model has become one of the most important models for the development of technical systems in recent decades.

1. Focused fields of application - Q: What is the field of application for the reference model? Are there special focuses or users?
2. Sequential or iterative approach - Q: Is the development process run in sequential or iterative order? Does the model point out the possibility of iterations?
3. Verification and validation - Q: Is the assurance of properties represented as forward or backward process?
4. Life cycle representation - Q: Does the approach refer to or include life cycle concepts?
5. Decomposition into System levels - Q: Does the model refer to different system levels? Are there hints at the decomposition of the system?
6. Adaptability for domains - Q: Is the approach adaptable to the different engineering disciplines or usable just for overall systems?
7. The requirement is traceable in each stage - Q: Is the requirement defined in the first stage will be traced and examed at each following stage?

8. Consider the agile concept - Q: Is the V-model adopt any agile concept in the product development process?
9. Consider the production and test stage - Q: Does the V-model consider the production and test stage for the product/system?
10. Consider the usage and service stage - Q: Does the V-model consider the usage and service stage for the product/system?

Table 1: Overview of characteristic properties related questions.

In this section, we will make a short evaluation and summary of 4 widely use V-model versions to demonstrate the different aspect of product or system development methods due to changes in technology, organizational and social environment. Before we start the evaluation, a few questions regarding the characteristic properties need to be elaborated and answered as shown in table 1 [13]. The main aspects we are going to explore in existing V-model are marked in gray from question 7 to 10.

Figure 1 (a) is the V-model of the VDI 2206 [33] guideline from the year 2004. For the first time, it offered a comprehensive model for engineering mechatronic systems. This V-model basically divides the development process into three sections: 1) The decomposition on the left side of the V-model describes the transformation of requirements, which are presented as an input, into a system design. 2) The engineering in different disciplines, the domain-specific design process. 3) Integration of the disciplines on the right side of the V-model during the system integration, verification, and validation. The result or output of the V-model is a product. Arrows between the two sides of the V-model illustrate the assurance of properties comprising verification and validation activities. Note, the assurance of properties is indicated right-to-left, which gives the impression of a retrospective process [13].

Figure 1 (b) is the extended V-model from Eigner. He combines the model for the approach in development with a virtual and data-based approach [7]. One of the highlights is the integration of a PLM backbone and the addition of an additional right wing for virtual testing which testifies a strong focus on model-based approach and virtual, hybrid or physical testing. Moreover, on the left wing, he adopts RFLP (Requirement, Function, Logic solution element, physical element) approach to composite the system design process. Comparable to Figure 1, in this V-model the "V" through the development process as well as the overall life cycle is shown.

Figure 1 (c) is the V-model by Bender 2005 [5]. This V-model has divided the system into 3 levels: system-level, subsystem-level, and component-level. It is noticeable here that the separation of the different domains does not take place at the top of the model, but already at the level of the subsystems. In addition, this V-model does not use any additional embracing to represent modeling and model analysis. This can be found in the description and of the individual sections. The model does not suggest an in or out track and ultimately does not suggest a product life cycle [13].

Figure 1 (d) is the V-model of INCOSE. It is illustrated as a sequential method used to visualize various key areas for SE focus, particularly during the concept and development stages. The "V" highlights the need for continuous validation with the stakeholders, the need to define verification plans during requirements development, and the importance of continuous risk and opportunity assessment [34].

All above-selected models are well established in their specific applications and frequently cited for scientific publications. As we summarized in table 2, these models have some features in common such as verification and validation process [13]. However, these V-models also varies in many different characteristic properties. For example, VDI 2206 focus on the application of the mechatronic system. On the other hand, Eigner's model is built up for MBSE and Bender' model focus on embedded systems. In our previous work, we have also adopted the V-model in the system development process of the digital factory and other use cases [24].

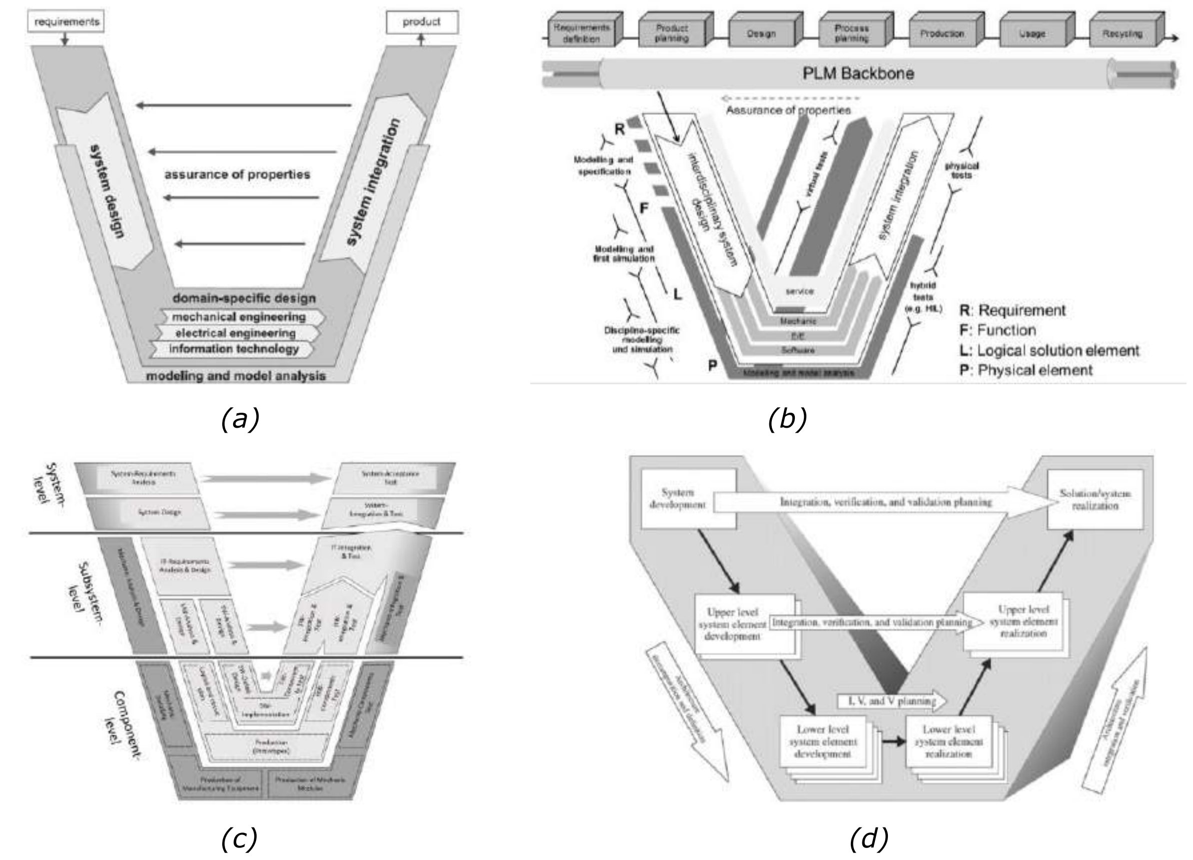

Figure 1: (a) V-model of the VDI 2206:2004 guideline "Design methodology for mechatronic systems" [33]; (b) Extended V-model for model-based systems engineering [7] (c) V-model by Bender 2005, translated from Bender (d) International Council on System Engineering [14] V-model [34].

As marked gray in table 2, we could see that current existing V-model versions didn't consider the agile concept, production and test stage as well as usage and service stage. Moreover, the requirement which is defined in the first stage is not able to trace and exam in the following steps of the model. Additionally, when we use V-model as the method of MBSE, we could notice there are many challenges such as 1) Non-uniform understanding of the overall system; 2) No consistency with regard to data, processes, and systems; 3) Component-oriented thinking; 4) Only discipline-specific models; 5) Inadequate methodology; 6) Lack of transparency and traceability of information. 7) A central system model for an interdisciplinary approach is not available; 8) There is a lack of formal and neutral languages, which are clear and interpretable. 9) Inconsistent relationship information. Therefore, in the following chapter, we adopt the concept of agile development method into MBSE and then proposed the new approach - Munich Agile MBSE Concept (MAGIC).

Characteristic properties	a) VDI 2206	b) Eigner,	c) Bender	d) INCOSE SE4.0
1. Focused fields of application	Mechatronic systems	MBSE	Embedded Systems	Systems Engineering
2. A sequential or iterative approach	Hints for iterations	Iterative	Iterative	Sequential, straight forward process
3. Verification and validation	Right to left	Right to left	Left to right	Left to right
4. Life cycle representation	No	Yes, backbone	No	No
5. Decomposition into System levels	Not illustrated	Not illustrated	Yes, three levels	Yes, three levels
6. Adaptability for domains	Yes, very generic	Yes	No, explicit description	Yes, very generic
7. Requirement is traceable in each stage	**No**	**No**	**No**	**No**
8. Consider agile concept	**No**	**No**	**No**	**No**
9. Consider the production and test stage	**No**	**No**	**No**	**No**
10. Consider the usage and service stage	**No**	**No**	**No**	**No**

Table 2: Overview of V-models by characteristic properties.

3 MUNICH AGILE MBSE CONCEPT (MAGIC)

MAGIC approach derived from the shape of DevOps infinity loop model and consists of 6 levels: system goal and requirement level, system functional level, systems architecture and logical level, system behavior analysis and verification and validation, system production and additive manufacturing and test, system usage and service level. In the following parts, we will describe each level in detail as well as the application of the MAGIC approach.

3.1 Concept of MAGIC

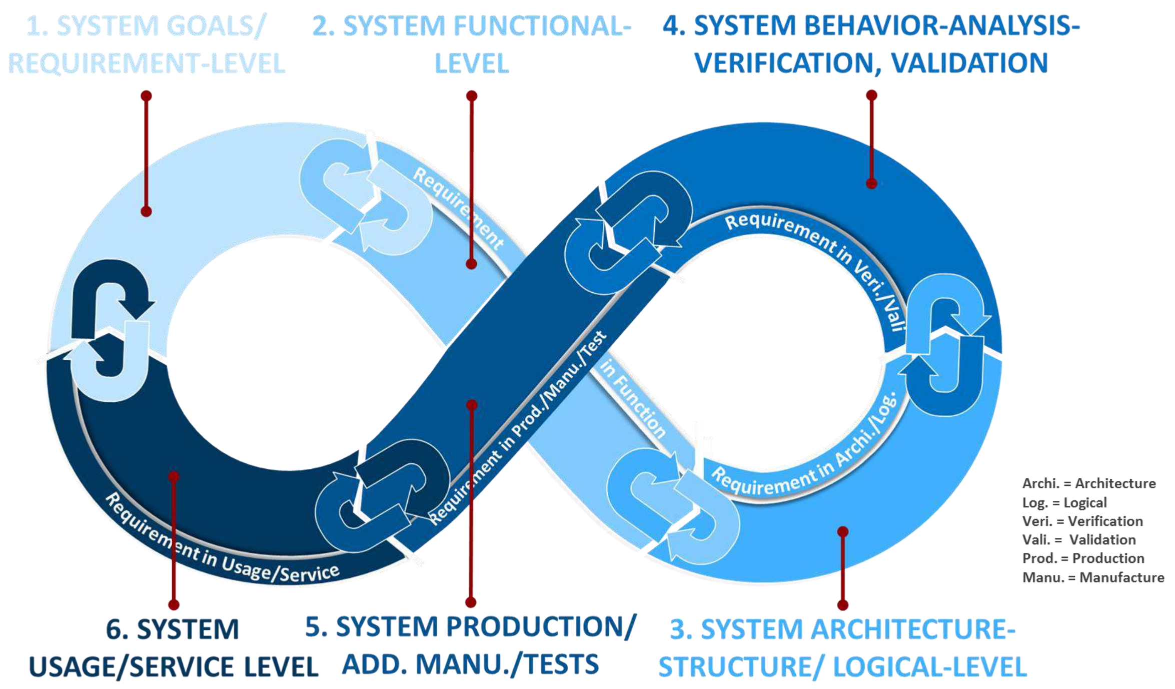
1. SYSTEM GOALS/ REQUIREMENT-LEVEL
2. SYSTEM FUNCTIONAL-LEVEL
4. SYSTEM BEHAVIOR-ANALYSIS-VERIFICATION, VALIDATION
Requirement
Requirement in Veri./Vali
Requirement in Prod./Manu./Test
in Function
Requirement in Archi./Log.
Requirement in Usage/Service
Archi. = Architecture
Log. = Logical
Veri. = Verification
Vali. = Validation
Prod. = Production
Manu. = Manufacture
6. SYSTEM USAGE/SERVICE LEVEL
5. SYSTEM PRODUCTION/ ADD. MANU./TESTS
3. SYSTEM ARCHITECTURE-STRUCTURE/ LOGICAL-LEVEL

Figure 2: The MAGIC approach according to Salehi.

1) **The system goal/requirement level** contains all the costumers need and wishes related to the product and its requirements. That means that the challenge during the first stages of the product development is to understand and to translate the costumers need into technical requirements. In product development and process optimization, a requirement is a singular documented physical and functional need that particular design, product or process must be able to perform. It is most commonly used in a formal sense in Systems Engineering, software engineering, or enterprise engineering. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system for it to have value and utility to a customer, organization, internal user, or another stakeholder. A specification may refer to an explicit set of requirements to be satisfied by a material, design, product, or service. In the classical engineering approach, sets of requirements are used as inputs into the design stages of product development. Requirements are also an important input into the verification process since tests should trace back to specific requirements. Requirements show what elements and functions are necessary for a particular project. This is reflected in the waterfall model of the software life-cycle. However, when iterative methods of software development or agile methods are used, the system requirements are incrementally developed in parallel with design and implementation.

2) **The system functional level** is a method for the understanding of the total product and to define defines the structure and behavior of a system. A system is structurally represented by the functional or structural relationships between the individual components or sub-functions with regard to the technical requirements. With the creation of functional system architecture, it is possible to identify the different functions in the entire system.

   - Abstraction of the overall function: the total function is derived from the request list

- Function hierarchy & structure: sub-functions are defined - functional decomposition
- Structure modeling: the sub-functions of the functional hierarchy are linked by means of material, energy and information flow. Once the functional structure is illustrated, the technical system architecture is created with consideration to the functional structure.

3) **System Architecture-Structure/Logical level** includes components which have logical ports with the logical behavior of a system. The technical system architecture of MAGIC approach describes a rough geometry, components design, and concepts, kinematics models. Simulate the logic models and modeling, the simulation tool used for the application example. It defines "how" the functional requirement will be achieved — and there can be multiple ways to achieve the requirements. The required logical architecture to provide functional services. On the logical level, the realization of the functions through the interaction of mechanic, electronic, and software components modeled. Different type of signals exchanged (e.g. control signals) as well as the type of persistent data. Analysis of the overall system behavior based on relevant parameters can be displayed quickly and clearly. In everyday language, verification is the answer to the question: Is a correct product being developed?

4) **System Behavior-Analysis Verification** means generally demonstrating the truth of statements. Transferred to technical systems, it is to be understood as meaning checking whether the way in which something is realized coincides with the specification. When checking the validity of a program, reference is also made to program verification. The verification is generally realized in a formal manner. On the other hand, validation is the answer to the question: Is the right product being developed?
System Validation originally means checking the validity of a measuring method in empirical social research, i.e. the extent to which test results actually register what is intended to be determined by the test. Transferred to technical systems, it is to be understood as meaning testing whether the product is suitable for its intended purpose or achieves the desired value. Validation comprises, for example, checking whether the description of an algorithm coincides with the problem to be solved. It generally does not have to be carried out in a formal manner.

   Verification is the "confirmation, through the provision of objective evidence, that specified requirements have been fulfilled." [15]. "Verification is a set of activities that compares a system or system element against the required characteristics. This includes, but is not limited to, specified requirements, design description and the system itself" [16]. Verification proves whether the system is created **right** [22]. Validation is the "confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled." [15]. Validation is the set of activities that ensure and provide confidence that "system is able to accomplish its intended use, goals and objectives in the intended operational environment." [16]. Validation proves whether the **right** system was created [22].

5) **System Production/Additive Manufacturing/tests level**: In order to achieve automation and data exchange with manufacturing technologies. In this stage, Industry 4.0 will be reflected. It includes cyber-physical systems, the Internet of things, cloud computing and cognitive computing. Production and logistics get right to the heart of the supply chain, from sourcing raw materials to delivering finished products to customers. Modern production and logistics methods are growing even more complex. Product lifetime is on the decrease, suppliers and buyers are spreading across the planet and more activities are being outsourced. In this level, system/product manufacturing and the test will be carried out. Additive manufacturing uses data computer-aided-design (CAD) software or 3D object scanners to direct hardware to deposit material, layer upon layer, in precise geometric shapes. As its name implies, additive manufacturing adds material to create an object. By contrast, when you create an object by traditional means, it is often necessary to remove material through milling, machining, carving, shaping or other means.

6) **System Usage/Service level**: In system usage/service level, we could conceptualize it in three dimensions - usage frequency, usage function and usage situation. The focus is the impact of these usage dimensions on different application scenario with customer satisfaction. System usage could include the following aspects:
   - Usage frequency: it refers to how often the product is used, regardless of the product functions used, or the different applications for which the product is used.
   - Usage function: it refers to what extent the product features/ functions are utilized by the consumer, regardless of how often the product is used.
   - Usage situation it refers to the different applications for which a product is used and the different situations in which a product is used regardless of either usage frequency or usage function.

3.2 3 Dimensions of MAGIC

The 3 dimensions of MAGIC as we can see in figure 3, are disciplines, tools, and systems, data and methods. These 3 dimensions describe the different aspects of MAGIC approach in the product development process. Furthermore, based on different disciplines (mechanics, electric/electronics, software), the MAGIC approach in each field of discipline could interact with each other in different stages through information flows. Moreover, it is noticeable that the portion of each discipline takes a different amount. As we know, from the last 50 years, the importance of software is increasing largely. Additionally, with the development of the electric and electronic component, the involvement of software in this discipline is also getting higher. Design and design methods of these 3 disciplines should be put to the test and their suitability for a modern interdisciplinary design approach should be checked and integrated into an interdisciplinary Method and process. IT solutions are partially available, but require further development in terms of applicability and integration.

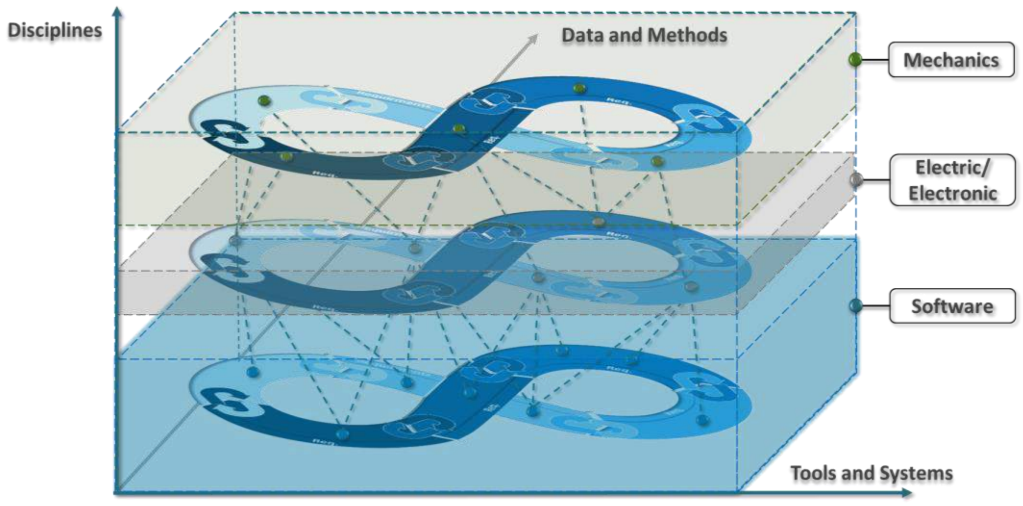

Figure 3: 3 dimensions of the MAGIC approach according to Salehi.

3.2.1 Disciplines:

Mechanics: Mechanics is that area of science concerned with the behavior of physical bodies when subjected to forces or displacements, and the subsequent effects of the bodies on their environment. Almost all procedural models established for mechanics assume a product development process consists of four main phases: requirements/task clarification, planning design, designing and working out, detailing [9].

Electronic/Electric: The consideration of design methods in electrical engineering and electronics (E/E) gives a broader picture than in mechanics. Since 1) the very different applications in the field of electrical engineering and electronics; 2) The fundamentally different approach of the designer in electrical engineering and electronics with respect to mechanics, which includes a design level of the schematic design before the layout (layout). This schematic level already has simulatable functional-logical elements 3) The rapid technological change, especially in digital circuit design. Complexity and integration density are constantly increasing. The requirements for higher packing density, smaller structure size, performance, smaller space requirement, higher clock rate, and lower power consumption are increasing. 4) The evolution of automation techniques in terms of logical and physical design as well as verification [9].

Software: it can be defined as goal-oriented provision and systematic use of principles, methods, and tools for the division of labor, engineering and application of extensive software systems. Due to the high cost of creating and maintaining complex software, the development is based on a structured process or phase and process models. These models divide the development process into manageable, temporally and content-limited phases. The software will be completed step by step. The phases are closely interlinked throughout the development process. In practice, methods that sacrifice the multilevel of system analysis, system design/concept, and subsequent implementation and testing are also used. These methods are called agile software development [9].

3.2.2 Tools and systems:

Tools and systems are the means to handle, process and execute the data and information which generated or collected during the system development process. It is significant to select the appropriate tool or system to build up a running environment for a different kind of data and information process. A proper tool or system can simplify the working process and accelerate the working efficiency. The tool is necessary to support the generation of a system model. Basically, it provides a user interface to facilitate the generation of the model.

3.2.3 Data and methods:

During the carrying out each stage of MAGIC approach, the data has been generated or collected from the beginning to the end. The method describes how the data has to be processed and used. Undoubtedly, we need to keep the valid data and work further. Besides, those data would be helpful to trace back the requirement or even perform validation and verification process.

3.3 MAGIC with MBSE

One of the core advantages of MBSE is transferring the document-based storage of data into model-based storage of data. In this way, the data can be accessed easier and the dependencies between data are more transparent [12]. Therefore, the transparency of development steps, as well as the traceability of changes increases and the handling of complexity, improve [19]. The application of MBSE is based on three pillars as illustrated in figure 4. System modeling by different roles and different teams establishes the foundation of MBSE. Methods, modeling languages, and tools are three crucial enablers for MBSE.

The method describes how the language has to be used to generate the actual system model. Previously, Estefan (2008) performed an extensive survey of six MBSE methods, which have been practiced in different industry projects [10]. Since then, new methods, in our case-MAGIC approach, have also been developed by various organizations for their internal exertion of MBSE approach [11].

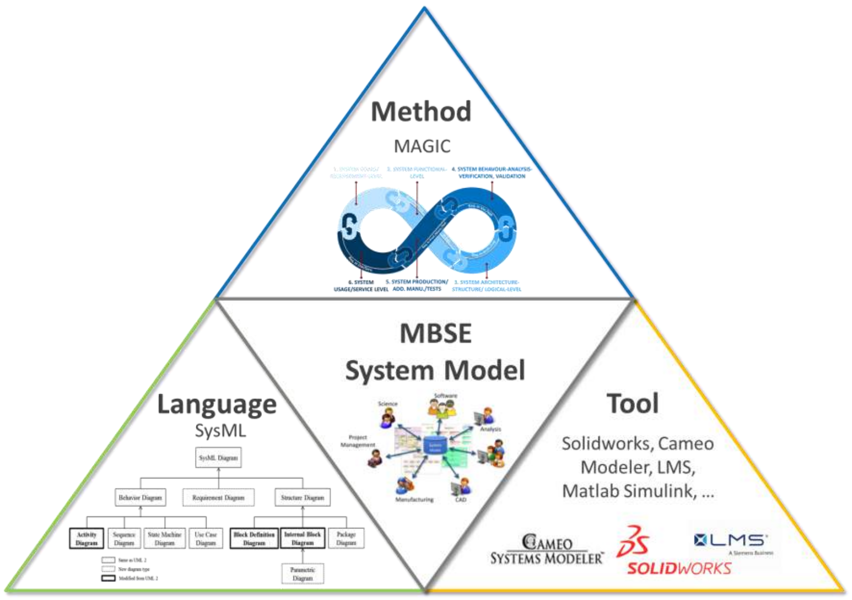

Figure 4: 3 Pillars of MBSE with the MAGIC approach as 'Method' according to Salehi

The graphical modeling language defines which elements and relationships within a model. Commonly used modeling language includes SysML, UML (Unified Modeling Language), IDEF (Integrated Definition Methods), etc. It could help users with specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities (Object Management Group, 2017b). A standardized and formalized language like OMG Systems Modeling Language (OMG SysML) is necessary for the formal definition of the system [11]. The origin of SysML is the Unified Modelling Language (UML) which was adapted by the Object Management Group Inc. to support the modeling of complete technical systems and integrating the needs of different domains. The concept of SysML consists of a central system model which can be displayed via different types of diagrams. Each of these diagrams offers a different view of the same system model and thereby supports different requirements to the view [20].

The tool is necessary to support the generation of a system model. Basically, it provides a user interface to facilitate the generation of the model. For instance, Cameo Systems Modeler is one of the tools to realize the modeling of the system. This MBSE environment of No Magic Inc. supports engineers with engineering analyses for design decisions, evaluation and requirements verification as well as checking model consistency and tracking design progress with metrics (Inc). Some other tools like Modelica, LMS, Solidworks, etc. are also been widely used.

4 USE CASE IMPLEMENTATION WITH MAGIC

In order to demonstrate the application of MAGIC approach, a use case – Mars rover development project has been introduced with a detailed step by step description.

4.1 EDMS Mars Rover

A Mars rover is an automated motor vehicle that propels itself across the surface of the planet Mars upon arrival. Rovers have several advantages over stationary landers: they examine more territory, and they can be directed to interesting features, they can place themselves in sunny positions to weather winter months, and they can advance the knowledge of how to perform very remote robotic vehicle control. There have been four successful robotically operated Mars rovers. The Jet Propulsion Laboratory managed the Mars Pathfinder mission and its now inactive Sojourner rover. It currently manages the Mars Exploration Rover mission's active Opportunity rover and inactive Spirit, and, as part of the Mars Science Laboratory mission, the Curiosity rover. On January 24, 2016, NASA reported that current studies on Mars by the Curiosity and Opportunity rovers will now be searching for evidence of ancient life, including a biosphere based on autotrophic, chemotrophic, and/or chemolithoautotrophic microorganisms, as well as ancient water, including fluvio-lacustrine environments that may have been habitable. The search for evidence of habitability, taphonomy, and organic carbon on Mars is now a primary NASA objective. Inspired by that, EDMS laboratory intended to design and develop a new Mars rover according to the MAGIC approach [24][25][25].

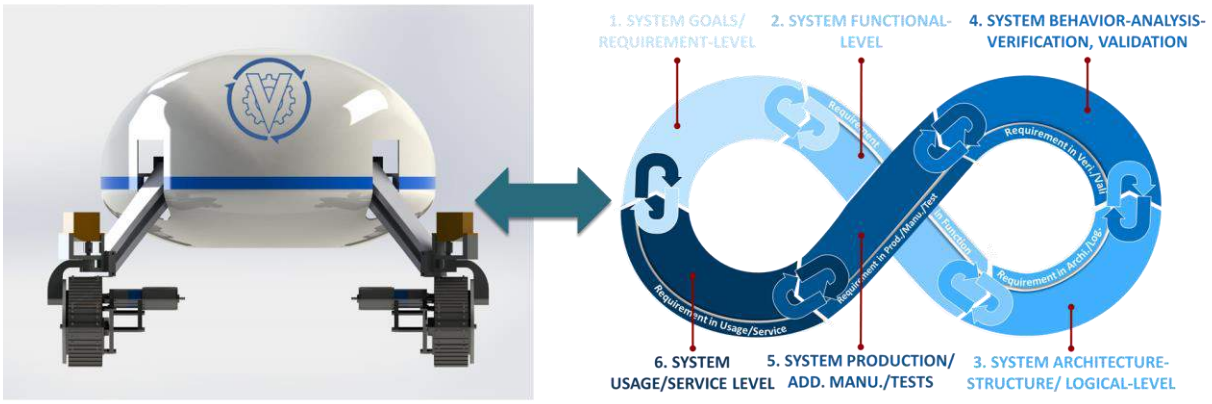
1. SYSTEM GOALS/ REQUIREMENT-LEVEL
2. SYSTEM FUNCTIONAL-LEVEL
4. SYSTEM BEHAVIOR-ANALYSIS-VERIFICATION, VALIDATION
Requirement
Requirement in Veri./Vali
Requirement in Prod./Manu./Test
in Function
Requirement in Archi./Log.
Requirement in Usage/Service
6. SYSTEM USAGE/SERVICE LEVEL
5. SYSTEM PRODUCTION/ ADD. MANU./TESTS
3. SYSTEM ARCHITECTURE-STRUCTURE/ LOGICAL-LEVEL

Figure 5: EDMS Mars rover implementation with the MAGIC approach.

Mars rover project is a proper example to apply MAGIC approach since it has been related to development regarding electronic, mechatronic, and software. Moreover, this project could also reflect well about the agile approach during the entire development process. The duration of the entire project is 6 months for 5 different team for dealing with tasks ranging from systems design, CAD design, production, product testing, and equipment integration. Each stage of MAGIC approach has been planned from 0.5 to 2 months." Also, within the engineering design the practices are well-known. But the implementation varies more in product development and is applied only to a certain degree. Tasks are in 3 teams of a more extensive character, duration could be several hundred hours, with a limited possibility of breaking down in smaller tasks. In this case, the sprint planning focuses on identifying the sprint goal, a subset of the completion of the task [26][28][29][29].

To everyday describe the progress for the fellow team members, make thus less sense. Another difference seen in the application is that tasks are typical of a more individual character. In for example engineering design, the developers to a larger extent have a certain area of the airplane as their responsibility. The allocation of a certain task to a certain engineer generally takes place already during the sprint planning in contradiction to software where this is allocated during the daily stand-up. The importance of not overloading a sprint is the same, but it comes to a more individual level of identifying an overload. The setting of priority is thus also done on a more individual basis. The focus it brings and with that minimum multitasking during a workday is central. Clear priorities and a balanced workload arguably boost efficiency and employee satisfaction wherever applied.

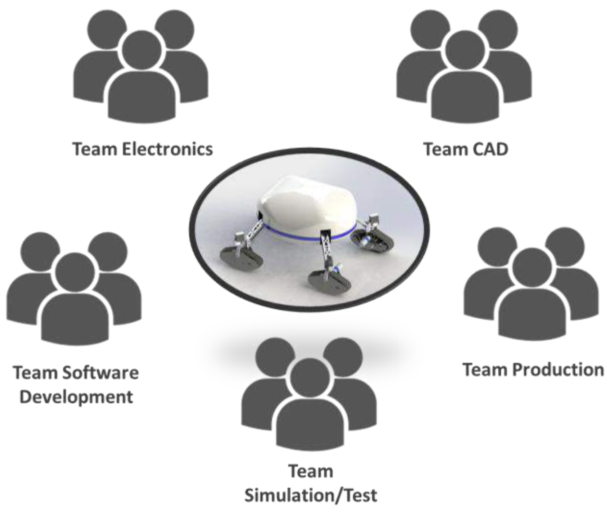

Figure 6: Different team with different roles.

4.2 Design & Development

In this section, we will follow each step of MAGIC and describe the design and development process of EDMS Mars rover in detail.

4.2.1 The system goal/requirement level

In order to cover each aspect of requirement, we have divided the requirement into 4 different categories: electronic requirement, mechatronic requirement, software requirement, and superior requirement. As shown in figure 7, Electronic requirement must consider batteries, control unit, and solar panels. When it comes to the mechatronic requirement, E-motor, servomotor, steering servo-motor should be included. Besides, for software requirement, we could separate it to embedded system requirement and mobile app requirement. Beyond all above-mentioned requirement, the superior requirement must contain all the points which should be considered first for the entire project.

4.2.2 System functional level

In the system functional level, the functions of Mars rover can be separated into 3 main functions, Mars rover movement, energy transformation, and signal transformation. As shown in figure 8, most importantly, the main function must reflect and satisfy the requirement. For example, function 'Mars rover movement' should satisfy the superior requirement 'Movement' with the detailed description: 1) 'The Mars Rover must have a driving mode and climbing mode'; 2) 'the Mars rover must have chain drive'. Under this function, the 'Mars rover driving control' should be included with height change function, initialization function, steering function, and velocity control function. Certainly, since adopting solar-panel is one of the requirements, in the system functional level, we need to include 'Photonic to electric energy' under 'energy transformation' function.

4.2.3 System architecture-structure/logical-level

In the system architecture level, we create a tree structure to demonstrate every single component of Mars rover and how it relates to others. In this case, as shown in figure 9 we design the Mars rover from the 4 main parts: body, autonomous system, chassis, electric/electronic. In order to highlight the electric and electronic component, we mark the block in yellow. It is important to mention that the bottom level of the tree structure is the blocks which represent CAD component associated with the CAD file.

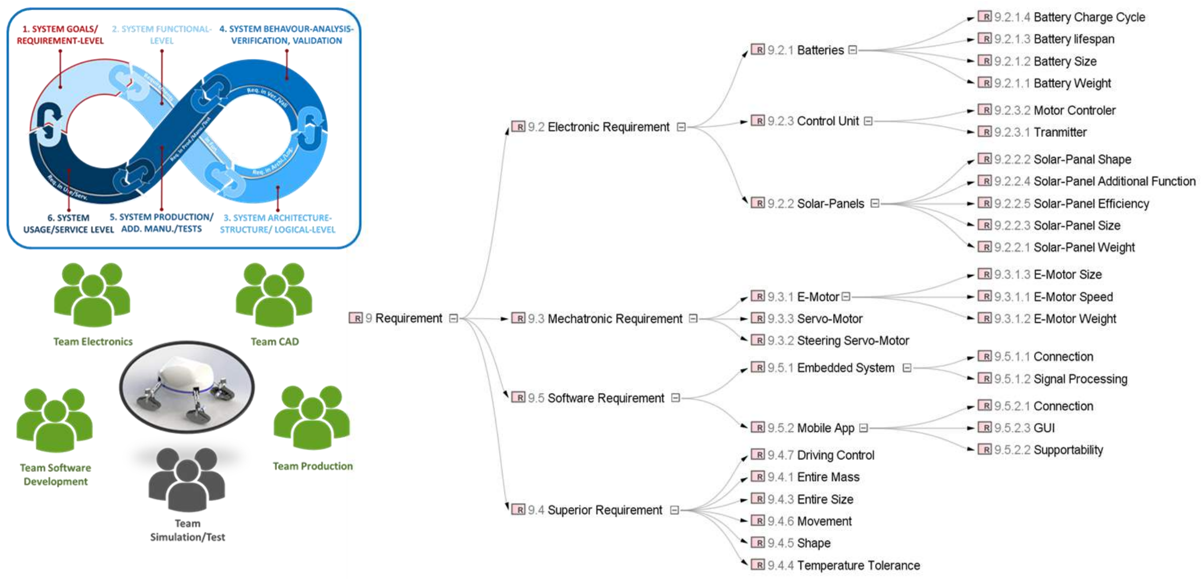

Figure 7: System requirement diagram associated with team involvement.

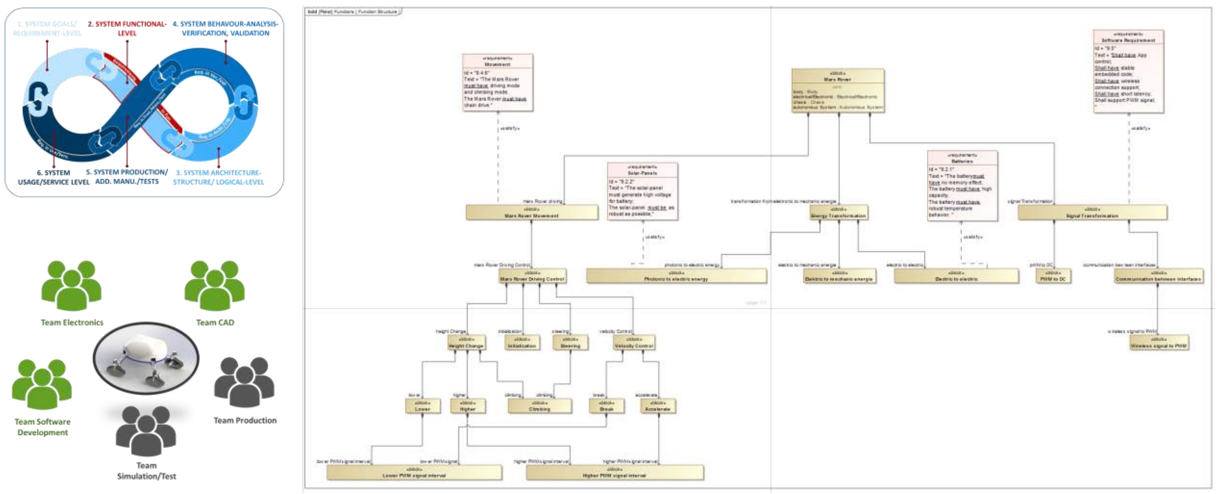
1. SYSTEM GOALS/ REQUIREMENT-LEVEL
2. SYSTEM FUNCTIONAL-LEVEL
4. SYSTEM BEHAVOUR-ANALYSIS-VERIFICATION, VALIDATION
6. SYSTEM USAGE/SERVICE LEVEL
5. SYSTEM PRODUCTION/ ADD. MANU./TESTS
3. SYSTEM ARCHITECTURE-STRUCTURE/ LOGICAL-LEVEL
Team Electronics
Team CAD
Team Software Development
Team Production
Team Simulation/Test
Mars Rover
Mars Rover Movement
Energy Transformation
Signal Transformation
Mars Rover Driving Control
Photonic to electric energy
Electric to mechanic energy
Electric to electric
PWM to DC
Communication between interfaces
Height Change
Initialization
Steering
Velocity Control
Lower
Higher
Climbing
Break
Accelerate
Wireless signal to PWM
Lower PWM signal interval
Higher PWM signal interval

Figure 8: System function diagram associated with team involvement.

In the block, we could also include the parameters of the component we have defined. Additionally, the system logic level should also include a state machine to demonstrate the different mode and stage of the Mars rover. For example, at the beginning, the Mars rover could be started and initialized. Then it will be in the waiting mode until it receives signals from the operator. Then it will change its mode to 'height change', 'starring change', 'stop' and 'speed change'. Finally, when it receives the 'stop' signal, the Mars rover will stop its movement and wait for the upcoming command.

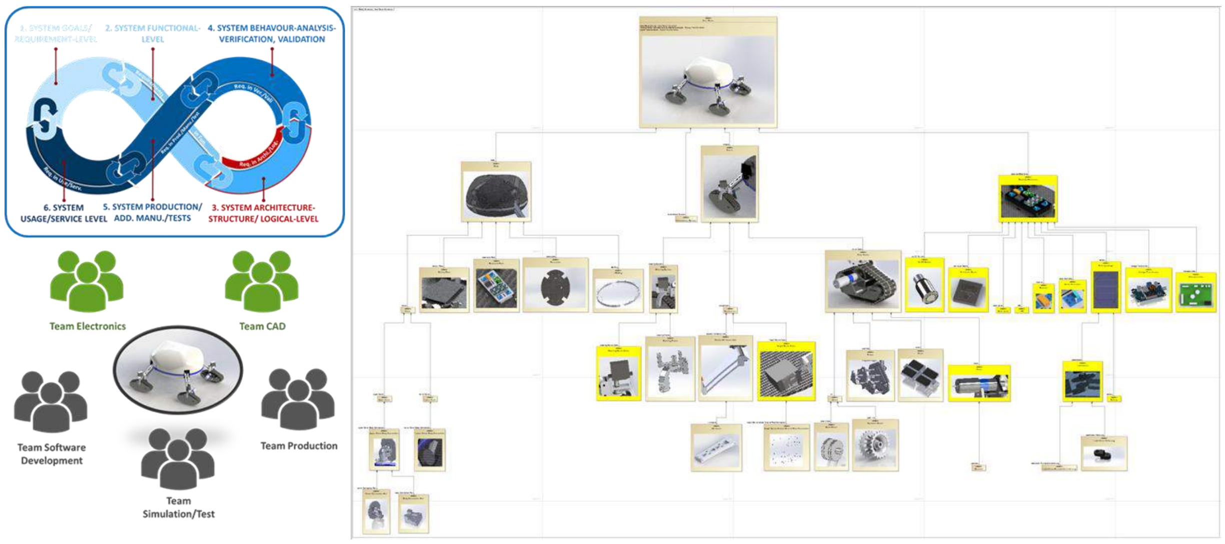
1. SYSTEM GOALS/ REQUIREMENT-LEVEL
2. SYSTEM FUNCTIONAL-LEVEL
4. SYSTEM BEHAVOUR-ANALYSIS-VERIFICATION, VALIDATION
6. SYSTEM USAGE/SERVICE LEVEL
5. SYSTEM PRODUCTION/ ADD. MANU./TESTS
3. SYSTEM ARCHITECTURE-STRUCTURE/ LOGICAL-LEVEL
Team Electronics
Team CAD
Team Software Development
Team Production
Team Simulation/Test

Figure 9: System architecture diagram associated with team involvement.

4.2.4 System behavior-analysis/verification & validation level

In the system verification and validation level, different kinds of simulations are necessary. For instance, simulations for digital security, checking the components, functionality test, and calculation of the physical values are the most significant factors which need to be verified and validate. Simulation tools such as Matlab Simulink, LMS could be helpful. In the simulation of driving mode, driving forward, reversing, speed up and down, and stop are 4 modes we care about most. As shown in figure 10, we could see that in this stage, team software simulation and the test will be involved.

4.2.5 System Production/Additive Manufacturing/Test-level

Production and manufacturing get right to the heart of the supply chain, from sourcing raw materials to delivering finished products to customers. Modern production and logistics methods are growing even more complex. Product lifetime is on the decrease, suppliers and buyers are spreading across the planet and more activities are being outsourced. In the system production level, we need to produce the component and integrate them into a complete functional Mars rover. In this stage, 3D printer is necessary. 3-D printing is an additive manufacturing (AM) technique for fabricating a wide range of structures and complex geometries from three dimensional (3D) model data [23]. For instance, as shown in figure 11, as a classic additive manufacturing process, once we have the detailed design table, we could set up the configuration and parameters in the machine and have the wishbone of the Mars rover. As modern 3D printing manufacturing, the designed CAD model will be directly sent and processed by the 3D printer and then print out the component as we designed. In this stage, team production and team simulation and test development will be involved.

4.2.6 System Usage/Service-level

In the system usage and service level, we take the real-life IoT (Internet of Things) scenario into account. As shown in Figure 12, while the Mars Rover is moving, different sensors installed on it will sense the environment and save the recorded data to the data logger. A data logger (also datalogger or data recorder) is an electronic device that records data over time or in relation to location either with a built-in instrument or sensor or via external instruments and sensors. Increasingly, but not entirely, they are based on a digital processor (or computer).

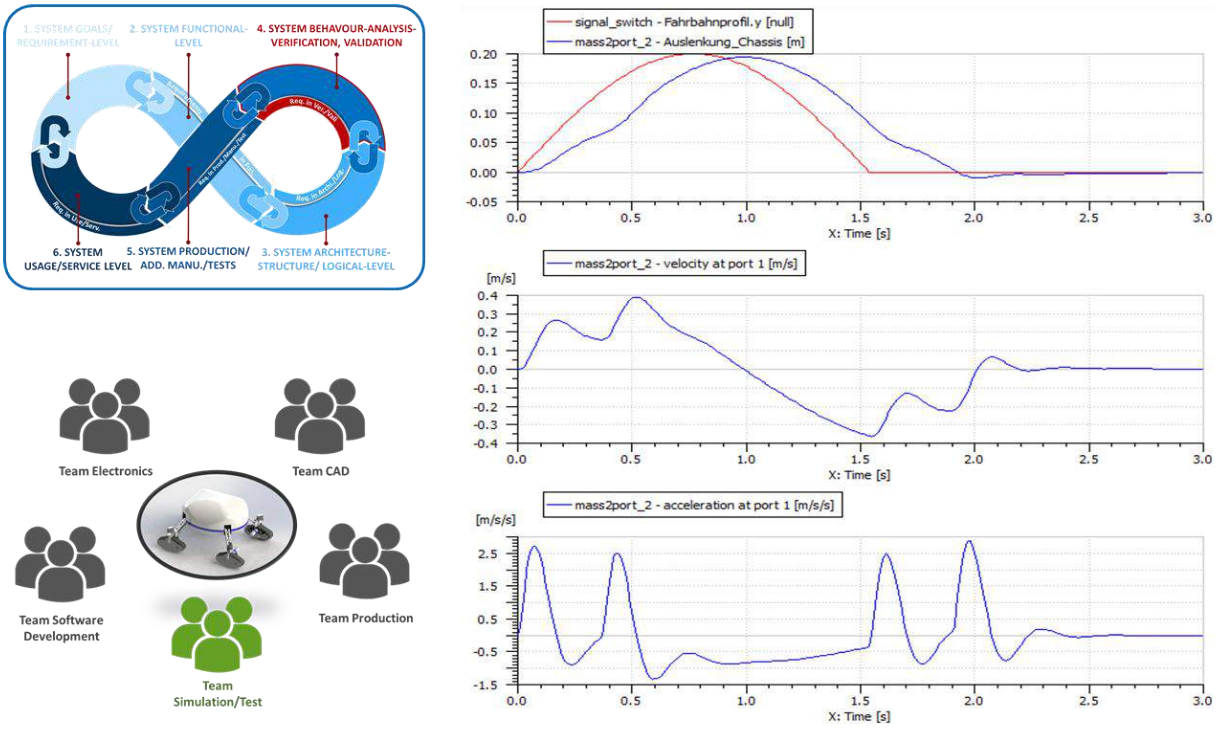

Figure 10: System verification & validation diagram associated with team involvement.

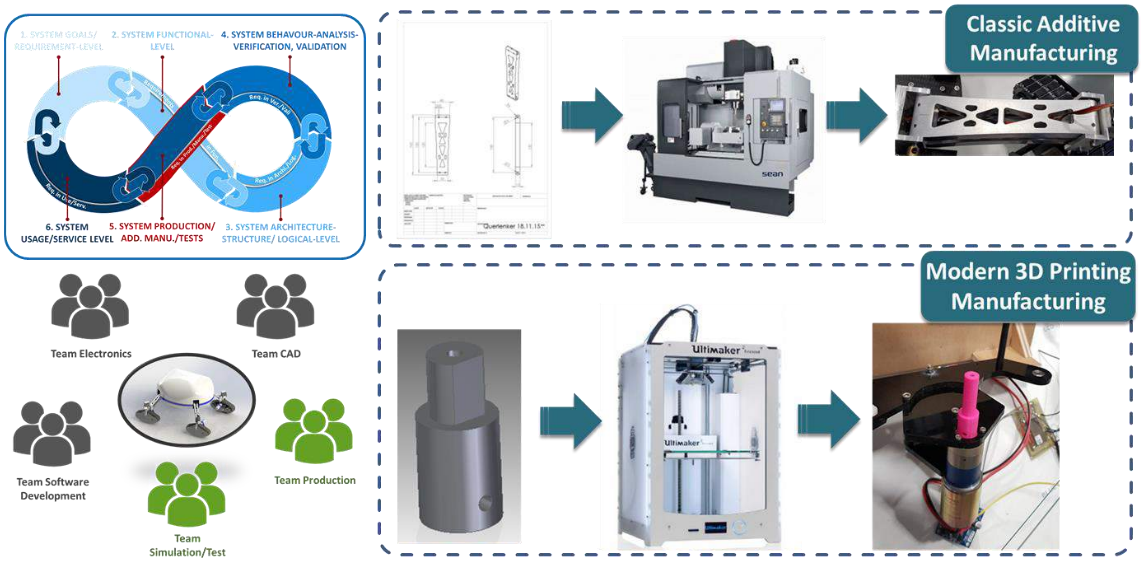
1. SYSTEM GOALS/ REQUIREMENT-LEVEL
2. SYSTEM FUNCTIONAL-LEVEL
4. SYSTEM BEHAVOUR-ANALYSIS-VERIFICATION, VALIDATION
6. SYSTEM USAGE/SERVICE LEVEL
5. SYSTEM PRODUCTION/ ADD. MANU./TESTS
3. SYSTEM ARCHITECTURE-STRUCTURE/ LOGICAL-LEVEL
Team Electronics
Team CAD
Team Software Development
Team Production
Team Simulation/Test
Classic Additive Manufacturing
Modern 3D Printing Manufacturing
Ultimaker

Figure 11: System production/additive manufacturing diagram associated with team involvement.

They generally are small, battery powered, portable, and equipped with a microprocessor, internal memory for data storage, and sensors. Some data loggers' interface with a personal computer, and use software to activate the data logger and view and analyze the collected data, while others have a local interface device (keypad, LCD) and can be used as a stand-alone device.

After the data has been saved, it could back up to the cloud service. After a certain data processing program, we could filter, analysis, visualize the data on the remote monitor such as a user App. In this stage, team electronics and team software development will be involved. This entire process is only one of the usage example of Mars Rover. Based on different requirement and situation, more usage and service could be provided.

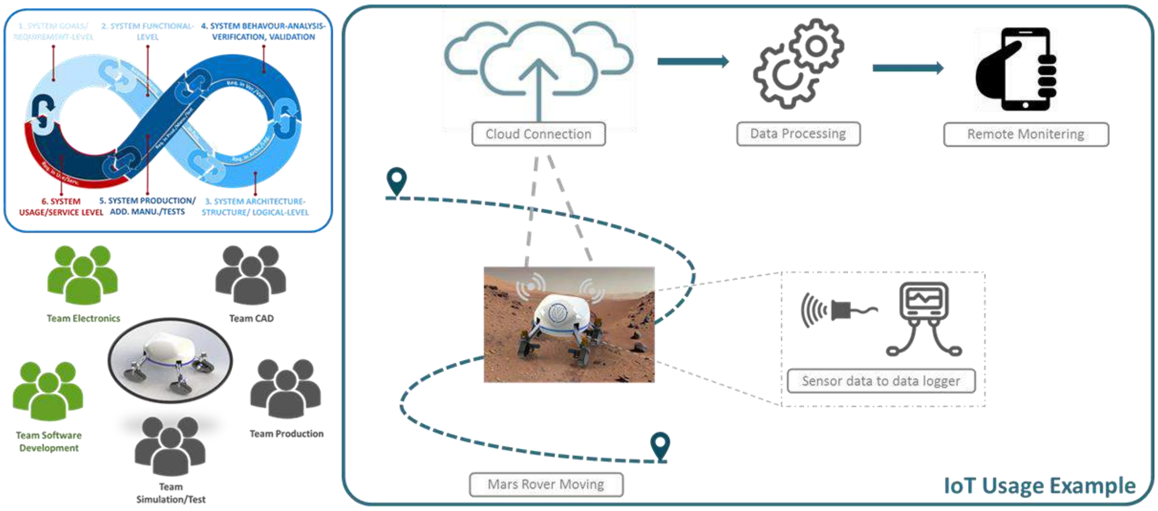
1. SYSTEM GOALS/ REQUIREMENT-LEVEL
2. SYSTEM FUNCTIONAL-LEVEL
4. SYSTEM BEHAVOUR-ANALYSIS-VERIFICATION, VALIDATION
Req. in Ver./Vali
Req. in Prod./Manu./Test
Req. in Archi./Log.
Req. in Use/Serv.
6. SYSTEM USAGE/SERVICE LEVEL
5. SYSTEM PRODUCTION/ ADD. MANU./TESTS
3. SYSTEM ARCHITECTURE-STRUCTURE/ LOGICAL-LEVEL
Team Electronics
Team CAD
Team Software Development
Team Production
Team Simulation/Test
Cloud Connection
Data Processing
Remote Monitering
Sensor data to data logger
Mars Rover Moving
IoT Usage Example

Figure 12: System usage & service diagram associated with team involvement.

5 CONCLUSION AND DISCUSSION

Model-based systems engineering (MBSE) is well-known in gaining the control over the complexity of systems and the development processes, while agile is a project management methodology originally from software development that uses short development cycles to focus on continuous improvement in the development of a product or service. In this paper, we adopt the concept of agile into MBSE and then proposed the new approach - Munich Agile MBSE Concept (MAGIC). MAGIC approach derived from the shape of DevOps infinity loop model and consists of 6 levels: system goal and requirement level, system functional level, systems architecture and logical level, system behavior analysis and verification and validation, system production and additive manufacturing and test, system usage and service level.

The highlights of MAGIC approach can be concluded as 1) the requirements which have been defined in the first stage will be examed and traced at each following stages; 2) communication between every 2 stages always exists in order to have a close connection during the system development phase. 3) the idea of Industry 4.0 has been included and reflected to achieve automation and data exchange with manufacturing technologies. 4) the concept of IOT (Internet of Things) is also considered when it comes to the usage and service of the system to satisfy the customer's needs. 5) the whole spirit of agile is reflected as the iterative and incremental design and development. Last but not least, a use case of Mars rover development project has been introduced to verify the feasibility of this approach. In order to verify the feasibility of this method, a use case of Mars Rover has been introduced.

Vahid Salehi, http://orcid.org/0000-0003-1577-5741

REFERENCES

[1] Albers, A.; Bursac, N.; Eckert, C. M.; Walter, B.; Wilmson, M. and Heimicke, J.: Agile method development: a live-lab case study on product properties for process planning. In: DS92: Proceedings of the DESIGN 2018 15th International Design Conference, 2018, 713–724. https://doi.org/10.21278/idc

[2] Anderl, R.; Eigner, M.; Sendler, U.; Stark, R.: Smart Engineering – Interdisziplinäre Produktentstehung – acatech diskussion, 2012.

[3] Bang, S.K.; Chung, S.; Choh, Y.; Dupuis, M.J.: A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps. RIIT. 2013. https://doi.org/10.1145/2512209.2512229

[4] Balzert, H.: Lehrbuch der Software-Technik, Lehrbücher der Informatik, 2. Aufl., Bd. 1. Spektrum Akademischer, Heidelberg. 2001.

[5] Bender, K.: Embedded Systems - qualitätsorientierte Entwicklung, Springer-Verlag, Berlin, Heidelberg. 2015. https://doi.org/10.1007/b138984

[6] Brown, T.; Wyatt, J.: Design thinking for social innovation IDEO, Development Outreach, 12(1), 2010, 29–43. https://doi.org/10.1596/1020-797X_12_1_29

[7] Cao, L.; Ramesh, B.: Agile software development: Ad hoc practices or sound principles? IT professional, 9(2), 2007. 41-47. https://doi.org/10.1109/MITP.2007.27

[8] Eigner, M.; Gilz, T.; Zafirov, R.: Interdisziplinäre Produktentwicklung - Modellbasiertes Systems Engineering. [online] PLM portal. Available at: https://www.plmportal.org/de/forschungdetail/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (accessed 12.07.2018). 2007.

[9] Eigner, M.; Roubanov, D.; Modellbasierte virtuelle Produktentwicklung, Springer Vieweg. 2014. https://doi.org/10.1007/978-3-662-43816-9

[10] Estefan, J.A.: Survey of Model-Based Systems Engineering (MBSE) Methodologies, INCOSE. Available at: www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf (Accessed 13.11.2017). 2007.

[11] Friedenthal, S.; Moore, A.; Steiner, R.: A practical guide to SysML, The systems modeling language, 3th Edition, Morgan Waltham. https://doi.org/10.1016/c2013-0-14457-1. 2009.

[12] Hooshmand, Y; Adamenko, D; Kunnen, S; Köhler, P. (2017) "An approach for holistic model-based engineering of industrial plants". Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 3: Product, Services and Systems Design, Vancouver, Canada, 21.-25.08.2017.

[13] I. Grässler, J. Hentze and T. Bruckmann V-models for interdisciplinary systems engineering international design conference - DESIGN 2018 https://doi.org/10.21278/idc.2018.0333

[14] INCOSE: INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition, John Wiley & Sons, Inc., Hoboken, New Jersey. 2015.

[15] ISO 9000; Quality management systems - fundamentals and vocabulary, 4th edition, ISO Copyright office, Geneva. 2015.

[16] ISO/IEC TS 24748-1 (2016), Systems and software engineering -- Life cycle management -- Part 1: Guidelinesb for life cycle management ISO Copyright office, Geneva.

[17] Jeon, B. W.; Um, J.; Yoon, S. C.; Suk-Hwan, S.: An architecture design for smart manufacturing execution system. Computer-aided design and applications, 14(4), 2017, 472-485. https://doi.org/10.1080/16864360.2016.1257189

[18] Kaufmann, U.; Schuler, R.: Systems Re-Engineering - ein Beitrag zur Integration von MBSE und PLM", Tag des Systems Engineering, Herzogenaurach, 25.-27. October 2016, Hanser Verlag, München, 343-352. https://doi.org/10.3139/9783446451414

[19] Kleiner, S.; Husung, S.: Model Based Systems Engineering: Prinzipen, Anwendung, Beispiele, Erfahrung und Nutzen aus Praxissicht", Tag des Systems Engineering, Herzogenaurach, 25.-27. October 2016, Hanser Verlag, München, 13-22. https://doi.org/10.3139/9783446451414

[20] Kößler, J; Paetzold, K.: Integration of MBSE into existing development processes - expectations and challenges". Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 3: Product, Services and Systems Design, Vancouver, Canada, 21.-25.08.2017.

[21] Lindlöf, L.; Furuhjelm, J.: Agile beyond software - a study of a large scale initiative international design conference, DESIGN 2018, https://doi.org/10.21278/idc.2018.0411

[22] NASA: Systems engineering handbook, NASA/SP, 2007-6105, Rev. 1, National Aeronautics and Space Administration, Washington, D.C., 83–106, 2007.

[23] Ngo, T. D.; Kashani, A.; Imbalzano, G.; Nguyen, K. T.; Hui, D.: Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. Composites Part B: Engineering, 143, 172-196, 2018. https://doi.org/10.1016/j.compositesb.2018.02.012

[24] Salehi V., Mcmahon C.: Action Research into the use of parametric associative CAD systems in an industrial context. In International Conference on Engineering Design, ICED'09, Stanford University, Stanford, CA, USA, 24 - 27 August 2009.

[25] Salehi V., Mcmahon C.: Development of a Generic Integrated Approach for Parametric Associative CAD Systems. In International Conference on Engineering Design, ICED'09, Stanford University, Stanford, CA, USA, 24 - 27 August 2009.

[26] Salehi V., Mcmahon C.: Methodological integration of parametric associative CAD systems in Product Lifecycle Management (PLM) enviroment. In: Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference DETC2009, pp.505-514. ASME, August 2009. https://doi.org/10.1115/DETC2009-86583

[27] Salehi V., Mcmahon C.: Development and Application of an Integrated Approach for Parametric Associative CAD Design in an Industrial Context, Journal paper, Computer Aided Design and application, 8(2), 2011, 225-236. https://doi.org/10.3722/cadaps.2011.225-236

[28] Salehi V., Mcmahon C.: Development of an evaluation framework for implementation of parametric associative methods in an industrial context. In International Conference on Engineering Design, ICED'11, DTU University, Copenhagen, Denmark, August 2011.

[29] Salehi, V.; Wang, S.: Using point cloud technology for process simulation in the context of digital factory based on a systems engineering integrated approach. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 3: Product, Services and Systems Design, Vancouver, Canada, 21.-25.08.2017

[30] Salehi, V.; Gross, F.; Taha, J.: Implementation of systems modeling language (SysML) in consideration of the consens approach Proceedings of the DESIGN 2018 15th International Design Conference 2987-2998, 2018. https://doi.org/10.21278/idc.2018.0146

[31] Schwaber, K. (February 1, 2004). Agile Project Management with Scrum. Microsoft Press. ISBN 978-0-7356-1993-7.

[32] Schwaber.K, Sutherland. J. The Scrum guide, 2001.

[33] VDI (2004), VDI 2206 - Design methodology for mechatronic systems, The Association of German Engineers (VDI), Düsseldorf.

[34] Walden, D.D.; Roedler, G.J.; Forsberg, K.; Hamelin, R.D.; Shortell, T.M.: Systems engineering handbook: A guide for system life cycle processes and activities, 2015, 4th ed., Wiley.

[35] Weilkins, T.: Systems Engineering mit SysML/UML: Modellierung, Analyse, Design, dpunkt Verlag, 2008, 2nd. Edition, Heidelberg, Germany.

[36] Zhang, T.; Dong, H.: Human-centred design: an emergent conceptual model, Include2009, Royal College of Art, April 8-10, 2009, London Include2009 proceedings (ISBN: 978-1-905000-80-7) available from http://www.hhc.rca.ac.uk/2084/all/1/proceedings.aspx