Requirements Eng (2014) 19:1–26
DOI 10.1007/s00766-012-0154-3

ORIGINAL ARTICLE

CORAMOD: a checklist-oriented model-based requirements analysis approach

William Brace · Kalevi Ekman

Received: 22 July 2011 / Accepted: 5 April 2012 / Published online: 25 April 2012

Abstract Requirement development activities such as requirements analysis and modelling are well defined in software engineering. A model-based requirement development may result in significant improvements in engineering design. In current product development activities in this domain, not all requirements are consciously identified and modelled. This paper presents the checklist-oriented requirements analysis modelling (CORAMOD) approach. CORAMOD is a methodology for the use of model-based systems engineering for requirements analysis of complex products utilizing checklists, the simplest kind of rational design method. The model-based focuses the requirements analysis process on requirement modelling, whereas the checklist encourages a conscious and systematic approach to identify requirements. We illustrate the utility of CORAMOD artefacts by a comprehensive case study example and modelling with system modelling language (SysML). We suggest that visual accessibility of the SysML views facilitates the full participation of all stakeholders and enables the necessary dialogue and negotiation. The approach promotes tracing derived requirements to the customer need statement and enhances validation by model execution and simulation.

Keywords Model-based · CORAMOD · Requirements analysis · Model-based requirements analysis · Requirement models · Formalize requirement

W. Brace (✉) · K. Ekman
Department of Engineering Design and Production,
Aalto University School of Science and Technology,
P.O. Box 14100, 00076 Aalto, Espoo, Finland
e-mail: William.brace@aalto.fi

1 Introduction

Inadequate development of requirements allows ambiguity to affect subsequent product development activities. This is usually evident during detail design and requires additional engineering and cost adjustment to unravel the missing or incorrect requirements. Requirements analysis (RA), one of the core engineering activities, continues to pose significant challenges. While very difficult, it is also extremely important because requirement shortcomings cause product failures and cost overrun in many projects [1]. Products are becoming more complex, and customer requirements are becoming more specific demanding greater value for a product. Customers have no interest in products or services per se; they look for solutions that they can use so that value is created for them. Deep understanding of customer needs [2], business and market needs, application domain and technology are fundamental to value creation, and this forms the basis of the future trend in requirement development.

The requirement development process often applied by engineering organizations is unstructured where every design agent operates autonomously (i.e. ad hoc approach) [3]. This empirical (experiential) approach is a document-driven development without a formalized modelling and collaborative process. There is therefore an increased interest of using systematic design methods. Many such methods exist pioneered by Pahl and Beitz [4] with the focus on engineering design, and Blanchard and Fabrycky [5] focusing on systems engineering (SE). Ulrich and Eppinger [6] laid more emphasis on providing better support for consumer products by considering industrial design aspects. All these methods have their differences, but they have at the core, a systematic design approach that emphasizes the importance of

requirement development. However, existing CAD/CAM/CAE systems and SE tools have major weaknesses. CAD/CAM/CAE systems currently provide no or little support for developing requirements. SE tools, on the other hand, support requirement development but do not consider, the systematic approach and existing knowledge on RA in the mechanical engineering design domain.

In recent years, various approaches jointly referred to as model-based systems engineering (MBSE) have been proposed to help address problems and difficulties in requirement development. Rather than just a new methodology, the application of MBSE in requirement development is a change of paradigm from document-driven requirement development process [7]. In this paradigm, requirement development is no longer a task for experts working in isolation or sitting around the table in group meetings. MBSE brings RA and modelling to the centre stage for use by all stakeholders. Following the systematic engineering design methods, we recognize the need to at least model requirements. A comprehensive modelling scheme is needed that enable engineering designers to analyse and generate requirements for their design, visualize how they are related and study the effect that changing an entity brings.

The objective of this paper is to present a novel MBSE methodology to formalize the RA in the engineering design domain by exploiting existing knowledge on requirement processes. The whole approach is considered as an extension of the systems modelling language (SysML) way of capturing requirements. Existing knowledge in the engineering design domain is exploited in previous works to propose the checklist-oriented requirements analysis (CORA) framework to address the challenges of requirement development. The CORA framework is proposed for the synergistic integration of the vast number of information and methods for the analysis of complex product requirements. The approach presented in this paper integrates the CORA framework with the SysML formalism to provide a solid requirement specification.

The remainder of this paper is structured as follows: there is the review of the state of the art in requirement development and model-based approach in Sect. 2. Section 3 reviews similar work to identify the novelty of the work presented in this paper. We present starting with the research context the proposed model-based process in Sect. 4 using a case study example in Sect. 5 to demonstrate its applicability. This is followed by a discussion of the research method and implication of the approach presented in Sect. 6. The last section summarizes and concludes the paper.

2 State of the art

2.1 Requirement development methods in engineering design

A complete engineering design process includes several phases. The first phase is conceptual design, which is the process by which the design is initiated and carried to the point of creating a number of possible design solutions [4, 8, 9]. The discrete activities considered under this phase are identification of customer needs and information gathering, problem definition, conceptualization, concept selection, refinement, and design review (Fig. 1).

Customers, the ultimate users or procures of products are capable of initially identifying the *need* for a product only in the most grand terms. The need statement may be a straightforward paragraph of one or more sentences that clearly define what the customer expects the product to do [10]. In other instances, it may be well documented into customer requirements and either of these is passed on to the next step (step 3). This step is characterized by activities, which serve to *refine* and *extend* the needs to *design specification* (DS) for use during subsequent phases. Extension in this context is a step-by-step analysis to find elusive requirements. Pahl and Beitz [4] recommended following a checklist and creating a scenario of the complete product life cycle. Refinement is the process of making the requirements less abstract [4]. Therefore, in the refinement process, requirements are presented to include *criteria* with specific *qualitative* and *quantitative* forms. The quantified requirements involve data with numbers or magnitude (i.e. 1 m, 4 s, $\leq$50 mm, high) and qualitative involves data with permissible variation or special requirements (i.e. waterproof, corrosion-proof). These first three steps are considered as a separate stage in most design literature (e.g. [4, 11, 12]) and are referred to as the *task clarification* or *problem space*.

The next steps, referred to as the *conceptual* or *solution space*, involves generating a broad set of solution concepts (step 4) that potentially satisfy the customer requirement. Team-based creativity is the main activity, and the best concept is selected through evaluation (step 5). To ensure that the solution concept is in line with the design specification, the designer(s) has to revisit step three. Therefore, not meeting some of the requirement criteria will require high changing and costly efforts. Furthermore, there is the need for more changes as requirements evolve at the conceptual and design phases.

2.2 Comparison of requirements development in software and design engineering

The starting point for the design of a software system and for an engineering design product is identified to be very

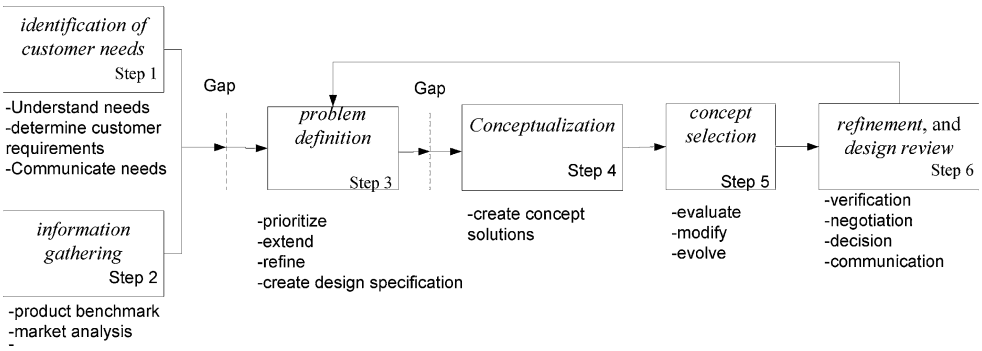

Fig. 1 The conceptual design phase

similar, since the task at this stage is the same (i.e. the expression of customer needs in relation to the real world) [13]. Furthermore, transformation of requirements from the informal to the formal is comparable suggesting that the processes will be alike. However, the close inspection of developing requirements differs or at least the terms do where the development is discussed, making the relevance of one to the other difficult to apprehend. Darlington [13] has developed the comparison of the salient differences in RA between the two domains (Table 1).

In essence, the tasks and vocabularies used are different. For instance, the formalization process as indicated in the last entry in Table 1 is dissimilar. In software engineering, the requirement development process is continued, where the description of the problem is transformed into a description of the solution. This is by way of successive re-description in different language formalism as indicated in Table 2. A similar approach of language re-description is applied in engineering design but is confined only to semantic re-description. Notwithstanding, there are four language representations (i.e. semantic, graphical, analytical and physical) for design purposes [14]. The design of a product is the refinement from abstract representations (need, goal or function) to a final physical product (form or structure). However, the requirement development process (Table 2) resides only in the use of textual language (semantic) alone. A similar approach of re-description of the different languages of the domain can be applied to make the formalization process result in part of the solution (i.e. creating function structure and behaviour models) by narrowing the solution space. The inclusion of the graphical and analytical languages in RA is possible and will narrow the gap between the problem and solution spaces.

In the software engineering domain, various RA methods appeal to the function, behaviour and structure model in an iterative process [10, 15]. This method is based on the logic of 'function follows form' with behaviour as an intermediary. The modelling approach is based on the notion of expanding the needs to expose functionality and allocating directly to physical forms of the product whilst attempting to understand the behaviour. Therefore, a sharp borderline between defining requirements and constructing the system design is difficult to draw [16]. This contrasts with the discrete nature of contemporary engineering design (Fig. 1). The concept of function, behaviour and structure has similar use. However, form–function relation is complicated and direct coupling between function requirements and products remains at an abstract level.

Table 1 Differences in engineering design and software engineering requirements development [13]

Engineering design domain	Software engineering domain
Customer needs refer to a mixture of concrete and abstract objects. However, the engineering solutions will always reside in concrete objects	Customer needs predominantly refer to abstract objects where the solutions also reside
There is no logical relationship between problem and solution mapping. The problem resides in semantics and the solution resides in the physical domain	Logical relationship between problem and solution. Both problem and solution reside in the domain of language
Formalization process substantially retains the use of semantics	Formalization process consists of re-description of the problem using different languages
Nature of the domain makes it difficult for attempts at proof-supporting formalisms	Logic structure of domain problems promotes the use of formalisms that provide automatic consistency, completeness, and ambiguity checking
Practitioners are accustomed to the use of semantics for description of domain objects	Practitioners are accustomed to the use of formal languages for the description of domain objects
Formalization does not result in any part of the solution	The formalization process results in description of solution

Table 2 Customer needs transformation to solution in terms of language usage [13]

Software engineering domain	Engineering design domain
Natural language	Natural language
Constrained natural language	Constrained natural language
Pseudo-natural language	Formally structured constrained natural language
Formal descriptive language	
Pseudo-code	
Formal code	

Mckay [17] claims that software, and electronic products differ from mechanical products due to the influence of the *physical effects*, *geometric* and *material characteristics* on the functionality. The behaviour can be defined as the way the physical process (i.e. physical, chemical or biological state) of the system evolves with time. According to Pahl and Beitz [4], the physical process is realized by the selected physical effects and its geometric and material characteristics. Physical effects can be described quantitatively by physical laws governing physical quantities and qualitatively by dimensions governing the physical quantities. Evaluation in the solution space of contemporary design is based extensively on the physical process formalize through the analytical language (i.e. equations, rules) based on the physical laws. The graphical language is used in traditional CAD tools for defining the geometric and material characteristics for manufacturability [14]. In addition, they are used in CAE tools for the modelling of the analytical language to solve the discrete and symbolic mathematical problems (i.e. 2D simulation diagrams). The formalization process in engineering design (Table 2) can be continued by including the analytical and graphical languages for a formal description. However, due to the abstract nature of the RA process, the analytical language formalism should be based on dimensions governing the physical quantities of the requirements rather than the physical laws, which are unknown at this stage.

2.3 The requirements modelling process

The following definitions for requirements and RA are adopted in this paper. A requirement is an essential *attribute* or characteristics defined for a product prior to efforts to develop a design. Requirements analysis is a structured, or organized, methodology for identifying an appropriate set of resources to satisfy the need and the requirements for those resources that provide a sound basis for design [18]. For a system in operation on its normal environment, the subsystems interact with each other and with the environment via *interfaces* in accordance with a predefined process to achieve the system goal. Therefore, a system is defined as the sequenced process that maps the product of the sub-systems, interface, and environment to a *function* [10].

This definition is used as a reference in developing systems in an inverted way starting with the highest level function which is the *customer needs*. This need is the *ultimate requirement* and all other requirements for the system in theory are or should be derived from it. This forms the principal axiom for the method described in this paper. Any requirement not traceable to the need may be a source of unnecessary costs [10, 18]. The basic process followed breaks down the ultimate requirement through a *systematic* process.

A systematic RA is also known as requirements engineering (RE). Requirements engineering covers multiple intertwined but well-defined activities. Several taxonomies have been proposed for the RE process based on these activities [19–21]. A general RE activity includes elicitation, analysis and negotiation, documentation, validation and management [22]. The RE process is, however, diverse in aspect and has different granularity. Furthermore, the approach is from software and systems engineering point of view. Nuseibeh [23] proposed a structure that includes elicitation, modelling and analysis, communication, agreeing and evolving. Davis [24] proposed a structure that includes elicitation, solution determination, specification and maintenance. A comparison with the general RE process is straight forward. Validation is included in maintenance, and management is implied in all the activities. In this paper, the process is structured as just requirements elicitation and requirements analysis. In comparing with the general RE process, management is implicit in all the activities while analysis, negotiation, validation and documentation are included in the requirements analysis. This structure is adapted to match the focused approach of the conceptual design phase (Fig. 1).

The approach is as shown in Fig. 2 and in this context, analysis modelling is the use of a combination of semantic (i.e. text), analytical (i.e. dimensions) and graphical (i.e. diagrammatic forms) languages for developing requirements. Therefore, the RA is a process where requirements are analysed, modelled, verified and managed to resolve possible conflicts [25]. The analysis is through a systematic use of decomposition, abstraction, projection and modularity to address the complexity by making the problems simpler. Outputs from the elicitation activities act as an input to the analysis phase. It is not the intention of the authors to elaborate on requirement elicitation activities in this paper.

2.4 A review of model-based systems and techniques

Model-based design, in contrast to document-based design, refers to a design process, where the documentation, the

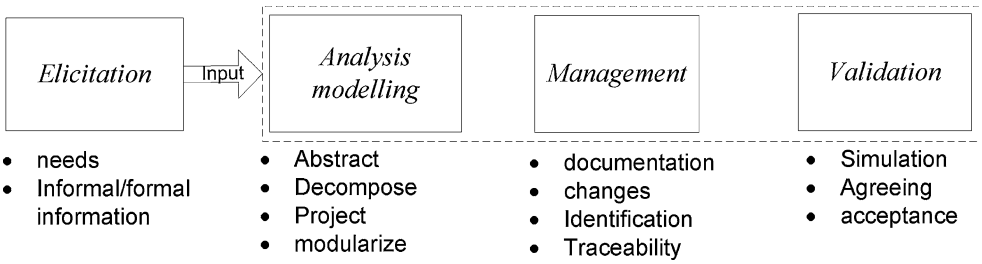

Fig. 2 The requirement analysis process

illustrations of the system and requirements are conveyed through the medium of a model [26]. A model-based approach has been the standard practice in the engineering design and other discipline for several years. However, this is applied in the later phase accompanied by physical experiments to evaluate the design. Nevertheless, with the increase in product complexity, engineering design has become simulation-based as physical experiment has become tedious and costly. Modelling and simulation enable designers to test the design specifications by using virtual rather than physical experiments. Most engineering analysis procedures require a complete description of the design product to be analysed. This makes simulation analysis applicable only during the later design phases when there is a physical model of the system. However, a much greater need exists for better analytical tools in the early stages of design when critical decisions are made based on the highly abstract design and qualitative information [27].

Therefore, there is the critical lack of tool support for RA. Model-based systems engineering (MBSE) is one technology that has been developed for this purpose [7, 28]. Nevertheless, the approach is placed firmly in the software engineering domain where much emphasis is placed on functional requirements. This affects the type of tools created. A number of reasons for a model-based approach are listed in [26, 29]. These include the following: (1) enhanced communication, (2) requirement allocation, (3) digital validation through simulation, and (4) reduction in design iteration. Models are used to extract the main characteristics and relationships of a system to show its requirements and to allow a holistic view of the overall system.

The environment of the model-based methodology is a collection of three elements (Fig. 3). These elements heavily influence each other in practices with regard to working efficiently in the environment. The model process embodies the following such as workflow, diagram usage and model organization [30]. Workflow describes the main activities and the order in which they are performed. Diagram usage guidelines may be defined to optimize communication and learning among team members. An important area for guidance is the model organization to support a simultaneous use by multiple users. It provides a

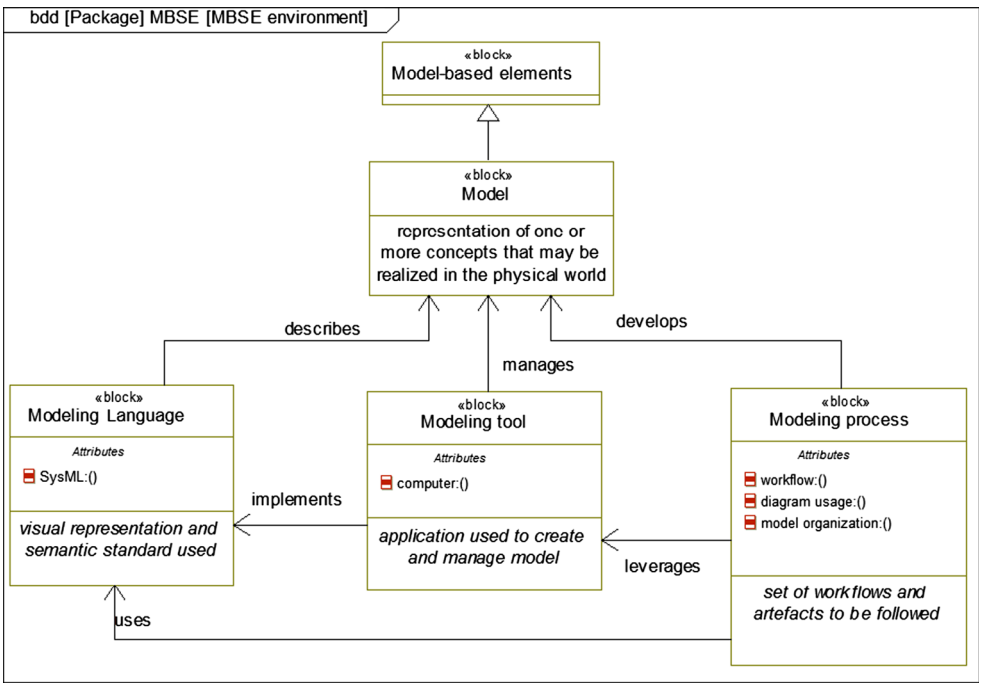

Fig. 3 Relation between elements of MBSE environment adapted from [30]

structure so that teams can work concurrently and effectively on a large model. Process in the context of this paper refers to a modelling process whose scope includes the use of systems modelling language (SysML™) in an MBSE environment.

SysML is a modelling language tool in systems engineering (SE). It is used in the model-based environment to define the visual notation (representation) and semantic (meaning) used to construct the model. SE is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed [5]. Fundamental to the application of SE is the system's life-cycle process. The V-model is a development model used to provide a life-cycle development template [31, 32]. It is a process that represents the sequence of steps in a project life-cycle development and is characterized by the iterative and incremental cycles through the analysis and design phase, the implementation phase, and the different levels of integration and testing.

SysML, which has its root from unified modelling language (UML™), has a standard taxonomy (Fig. 4). The standardized diagrams support specifications, analysis, design, verification, and validation of a broad range of complex systems [33]. In particular, the language provides graphical representations with a semantic foundation for modelling system requirements, behaviour, structure, and integration with a broad range of engineering analysis. Therefore, modelling in SysML serves as an integrating framework to support a system development environment (SDE). Contemporary design for complex products is supported by the SDE which refers to the computer-based, multiuser, networked tools and repositories (i.e. CAD, engineering analysis and simulation tools) used for system development [26]. SysML uses the Metadata Interchange (XMI®) to exchange modelling data between the various tools and is also compatible with other data exchange protocols (e.g. XaiTools) [34]. Therefore, in the simulation-based design, models can easily be transferred to other simulation tools for analysis and simulation.

Overall, SysML is designed as a general-purpose language. Conversely, practitioners want to solve specific problems. The challenge is to define a modelling process that supports the intended goal. In this paper, we define a formal RA procedure that can be included in SysML. This approach is possible because of the large flexibility of SysML that allows customization.

3 Related work

The work reported in this paper overlap with requirement development in the engineering design and RE, and modelling process in software and systems engineering. From the review of literature, several different approaches exist for including requirement development into engineering design. However, the review also indicated that very few computer-aided engineering tools include RA in the system.

A common approach used is the traditional document-based method. The designer has to go back and manually refer to the design criteria in the specification whenever the need arises (Fig. 1). One method of particular used extensively for this approach is quality function development (QFD). QFD requires that the customer needs are quantified and translated into requirement specification and subsequently measured against how well the customer need

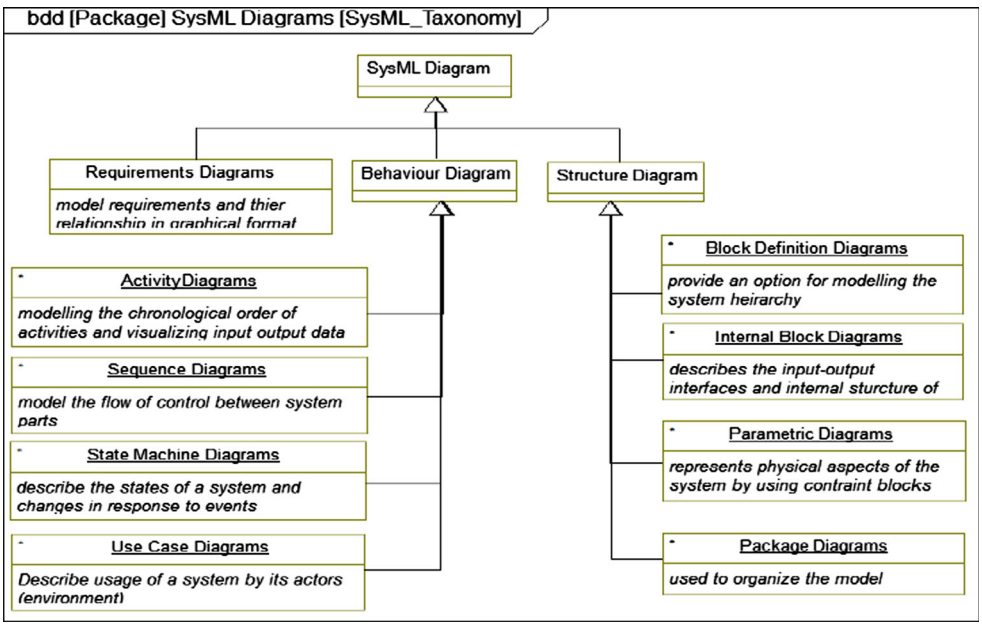

Fig. 4 Diagram of SysML taxonomy adapted from [33]

is satisfied. The aim of QFD is to improve the quality of design, and as such many publications are devoted to its application [35]. However, the model does not provide a graphical expression and simply establishes a relationship between information pairs [10].

The application of knowledge to design requirement development can be found in a number of investigations. However, most of the applications of existing tools are limited to mapping functional requirements to the solution space to generate concepts and thus ignoring other crucial requirements. The PROCONDES system developed by Rehman and Yan [36] map functional design specifications to the solution space and automatically generates other requirements in the solution space. Tseng and Jiao [37] developed a requirement management tool based on recognizing patterns of functional requirements from past design episodes.

Most of the research and practices in engineering design are motivated by the realization that better computer-aided design tools are required. Existing computationally based tools in the conceptual design phase aim to produce design automatically from the design specifications. Mamat [38] developed a system aimed at providing support during the design of automobile seats by integrating requirement development in an engineering CAD system (Autodesk Inventor). Therefore, the system incorporates requirement development in a computer-aided geometrical modelling (CAGM) environment. Nonetheless, only requirements related to the product usage (from a comfort point of view) and manufacturing issues were analysed. Galea with colleagues [39] emphasized the importance of incorporating RA in CAGM and developed a tool that considers all the product life-cycle requirements. However, in both approaches, requirements are developed from a chosen concept in the solution space, ignoring the systematic analysis of the design need in the problem space.

It is clear from above that the traditional approach moves from the problem space, to the solution space. However, the approach is document-based, and the specification is not mapped to the solution in the solution space [40]. The application of knowledge-based and computer-aided systems is rather in the solution space. This indicates the problem structuring processes is not necessarily a natural one for engineers. Many engineers are more solutions oriented and apply sorely creative processes and experience [12]. For a sufficiently simple problem, experience and creativity can be applied directly to the design problem in the solution space. However, for a complex product, where various specialists are involved, it is very risky to apply this approach.

Tools, which support RE, can be divided into modelling and validating tools and management tools [19]. Present-day requirement management tools, including SLATE [41] and DOORS [42] provide the best support for top-down development where the focus is on traceability and allocation. Modelling tools, on the other hand, enable the creation of graphic and textual models of the requirements and allow consistency checks. UML is a methodology-independent graphical notation for modelling. It can be used to specify, visualize and document requirement models [43]. However, the method is good for high-level and more detailed analysis, and specification of software requirements. SysML was developed to mitigate the challenges of UML and is used in engineering design. The main purpose was to mitigate the challenges confronted during the conceptual phase of system design (Fig. 1). Nevertheless, it concentrates more on the concept phase and marginally touches on the task clarification or problem phase. The analysis of requirements is therefore marginalized.

It is clear from the study of related works that very few design support tools take into account all factors needed to develop requirements into design specifications. The application of computer-aided design tools for including RA is mostly concentrated on the solution space. The novelty of our approach is the introduction of a *common conceptual model* for the problem domain and solution domain combined with step-by-step *conscious analysis* of requirements. Such gradual, tool-based transition from the customer requirement to design specifications reduces the gaps within the phases (Fig. 1) by narrowing the boundary of the solution space.

Conscious analysis in this sense is the use of the checklist structuring, classification and decomposition strategy. Checklist is the simplest form of rational method, which encourages a systematic approach to design. A conscious process is a process that is executed by people who are aware of all aspects needed and use this knowledge in analysing and reviewing. SysML is used to include the graphical language description in RA and as a management tool and to integrate the RA into the system development environment. The basic SysML requirement diagram is extended in our approach, by creating stereotypes and tagged values and by analysing and grouping related requirements. Therefore, the capability of SysML is exploited by using the standard taxonomies for problem analysis in the problem space. In addition, the taxonomy is used to create dimension networks based on the physical quantities to include the use of analytical language description in requirement modelling.

4 Checklist requirements modelling process overview

4.1 Research context

The proposed approach is as a result of an ongoing research project taking place in the context of adaptive design (i.e.

using established solution principles) [4] where an existing relatively complex system is being developed. The research project domain is the HybLab research network of the Aalto University. The Hyblab project is an industry-related project in an academic setting. The aim of the network is at researching on hybrid energy systems for heavy-duty work machines. The higher-level research goal of the project is to reduce the energy consumption into half of today's level. There is an investigation of an integrated design process due to the complexity of the research project. Therefore, the network is interdisciplinary in nature and consists of many specialties. Our intention is to make it possible to optimize the develop system from several points of view (e.g. technical, economic, manufacturing, environmental). Consequently, one of the objectives is to develop requirements and models and to formalize to guide in the design and simulation process.

The complex system is an industrial forklift truck system for loading pallet goods in warehouses. The research team is a collection of engineers from different faculties in the university and personnel from various industries. The physical product exists and is systematically dismantled to analyse and gain knowledge. At the same time, there were several information searches while other products are benchmarked.

In a typical complex product development in an engineering project, different groups of people are responsible for collecting market information, determining customer requirements and product benchmarking. The requirement engineer may not be involved in all these steps. However, in the context of this project, the method used includes participation in the project. The use of participation aims to gain a close familiarity with the practices through intensive involvement with the engineers in their natural environment over an extended period of time [44]. In addition, the strategy used in collecting data includes direct observation, informal interviews, collective discussions, analysis of personal documents, analysis of the existing physical product and technical documents. There was the further iterative review of the literature on RA in the engineering design domain. From the literature studies, it was evident there exists a disparity between the idealized representation of RA as presented by design methodologists and the responsive ad hoc approach in the real world as confirmed in the case studies by Darlington and Culley [3].

The starting point in the requirement development is the documented expression of the customer needs or market demands and may be presented to the requirement engineer as initial requirements in one of the following forms [4]: (1) as one sentence or in the form of short narrative statement and (2) as very detailed with a profusion of texts, graphs, or even formulas. In the context where the initial requirement is presented as in (1), which was the case in this project, the requirements need to be reviewed in order to extend and formalize to develop requirements as we illustrate in Sect. 5. The RA process in this context is the modelling and analysis of the initial requirement whose scope includes the use of checklist RA in an MBSE environment. There is the use of a computer modelling tool implementing the semantics and visual presentations defined by SysML to manage the process and to allow integration with other modelling and simulation tools. In the next sub-sections, we discuss the proposed RA process.

4.2 Requirement model organization

Systematic design methods suggest an integrated set of models and documents that provide guidance with regard to model organization. Model organization has a profound effect on the usability of the model as it grows over time and how it will work in a team environment. The model organization is recognized as a critical aspect of MBSE. Typically neglected, we believe that an integrated model will facilitate the RA process.

In this work, the RA process is based on the checklist-oriented requirements analysis (CORA) framework. In earlier work, we have identified this framework as a suitable base for model-based systems. The framework is used to organize relevant requirement information in addition to demonstrating a checklist decomposition and analysis of requirements. The organization of requirement information is not considered in this paper. The V-model is used as the process model. The framework is integrated into the V-model to create CORAMOD, the checklist-oriented requirements analysis modelling process (Fig. 5).

CORAMOD provides a seamless integrated modelling between requirement decomposition, analysis, and validation. It is important to note the creation and reuse of requirement-related checklist stereotypes (Fig. 6).

The stereotypes are used to assist in both the top-down structured decomposition, bottom-up integration, and validation of the requirements. Domain experts can also be allocated to each checklist stereotype. This is to allow for easy access to experts and to enhance collaborative work. Within CORAMOD, the key objectives of the model-based RA process are as follows:

1. Decomposition of the ultimate requirement
2. Identification, analysis and derivation of functional and non-functional requirements
3. Identification of dependencies between the requirements
4. Quantification of the derived requirements and validation

CORAMOD requires an internal organization that supports the simultaneous use by multiple users and

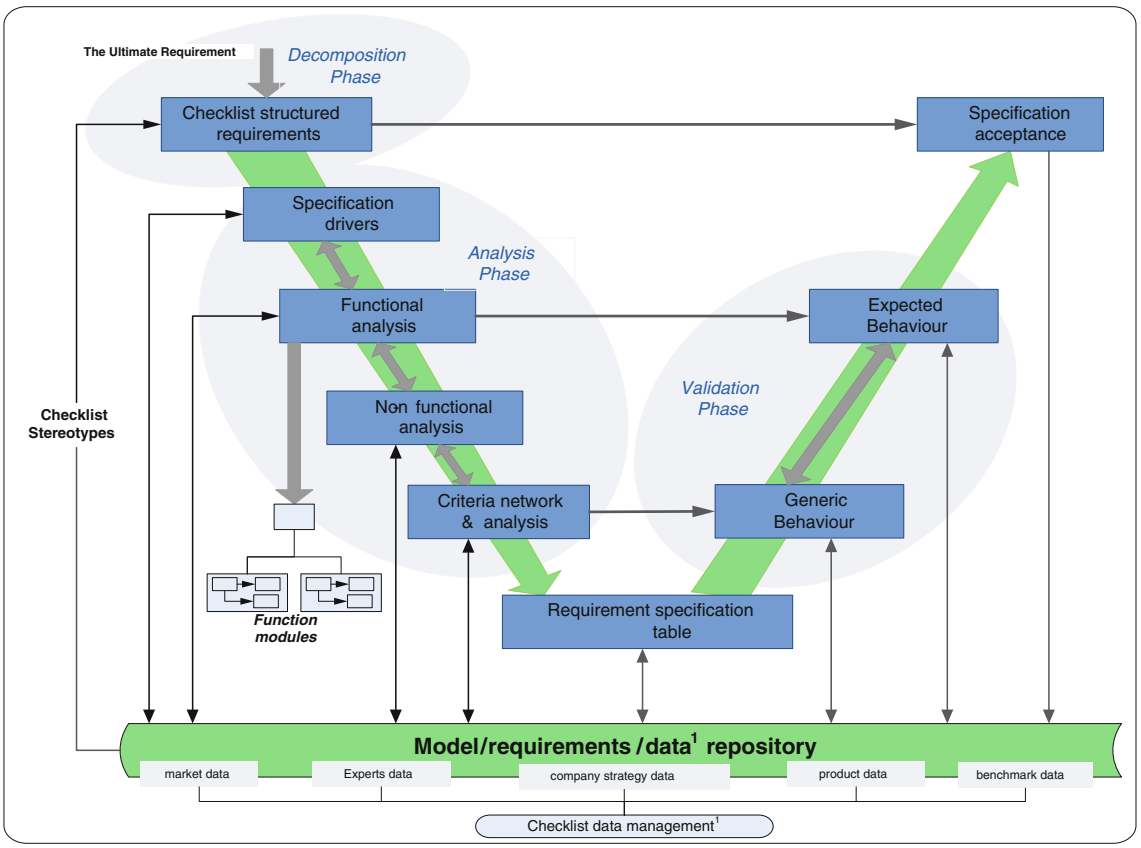

Fig. 5 Checklist-oriented requirements analysis modelling process (CORAMOD)

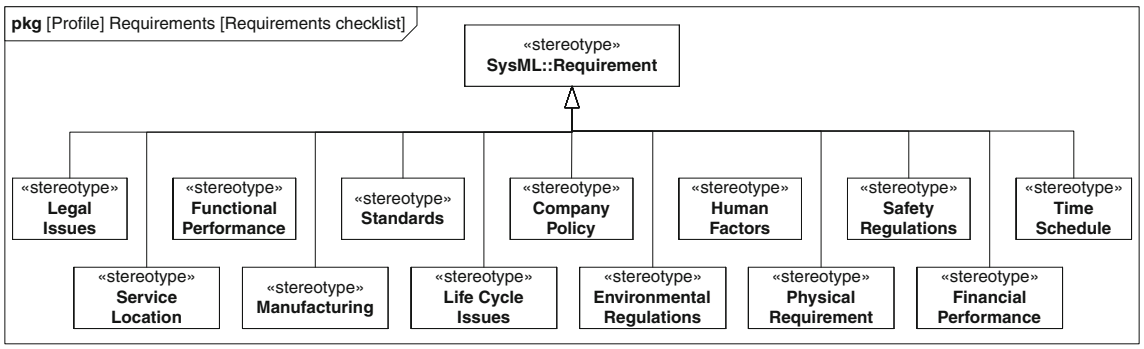

Fig. 6 A package diagram containing checklist stereotypes

integration with other analysis tools. Therefore, it leverages SysML as the graphical modelling language. Since simulation has become indispensable in contemporary engineering design, RA modelling in SysML provides an integrated system development environment to achieve modelling and simulation interoperability.

4.3 Modelling process workflow

CORAMOD in SysML (Fig. 7) is used to capture functional requirements and constraints (non-functional requirements) among the various aspects of the complex product and its environment. The main emphasis is to derive requirements with the simultaneous identification of the product functionality, behaviour, and performance values. The functionality is further abstracted as separate function modules to identify more requirements. The behaviour and performance values are modelled for simulation to validate the requirements and to refine the functionality. When completed, the function modules are allocated to a subsystem for requirement verification. Simulation and verification are not considered in this paper.

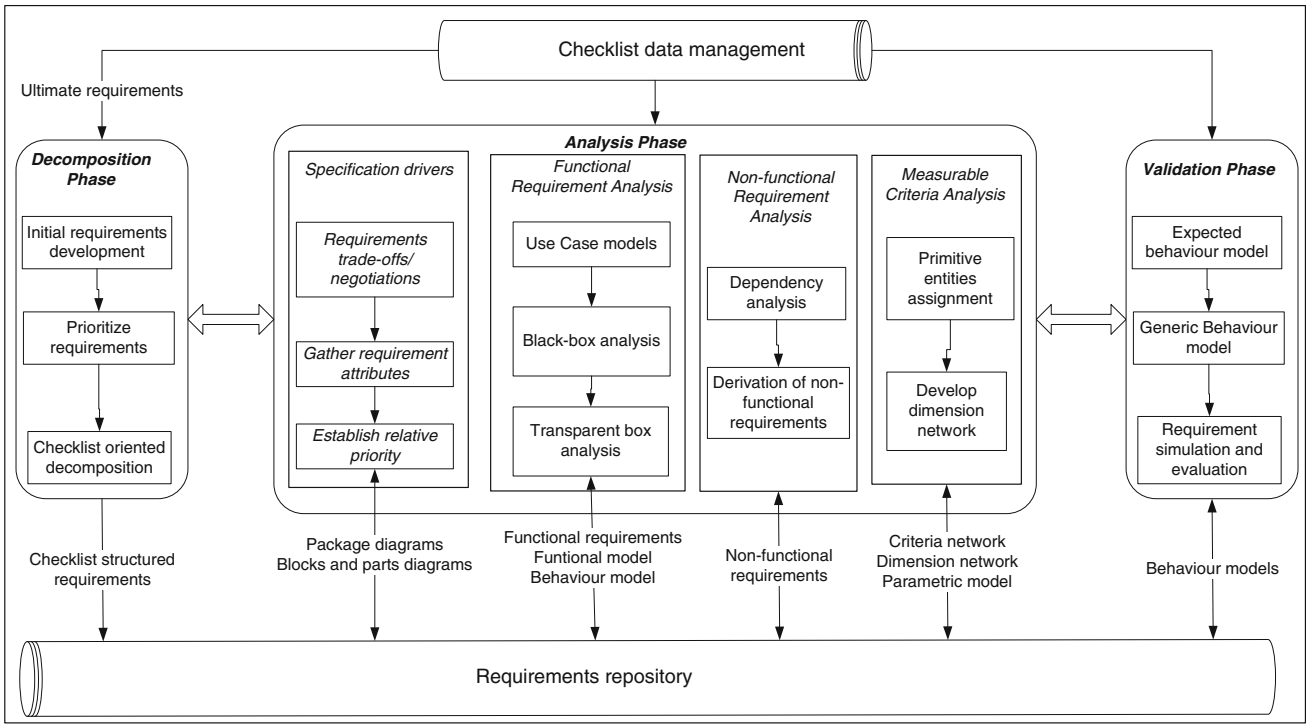

Fig. 7 CORAMOD in SysML workflow

A high-level summary of each phase (Fig. 7) is as follows. The essential task in the decomposition phase is deriving requirements from the customer statement and prioritizing as initial requirement statements. By discriminating the derived requirements under the checklist stereotypes, we then create a structured requirement.

Prior to the RA, the context for the analysis is set using the specification driver method. The specification driver is a platform for team negotiation and collaborative discussion. It provides a platform for stakeholders to work together to prioritize candidate requirements and to identify potential conflicts among requirements. The platform is also used to determine the requirement candidates upon which other requirements can be based.

In the analysis phase, requirements under functional-performance checklist are modelled as use cases (UC) and transformed into the coherent description of system functionality (i.e., create function structure). The analysis is then performed in two steps: black-box analysis and transparent-box analysis. The overall system functions are identified and validated in the black-box analysis. In the transparent-box analysis, the system-level function is logically arranged into sub-functions based on priority. These are further modelled into function modules and analysed. Requirements are derived and recorded in all these steps.

The emphasis in the non-functional requirement analysis phase is on the identification of dependencies and links between the various requirements in the checklist category and external CAGM environments. Non-functional requirement statements are very often unbounded, for example, improve safety. Therefore, they are restated to define specific bounds and analysed in order to derive requirements and transform them to be validatable.

The measurable criteria analysis is used as a basis to quantify the requirement and create networks based on the dimensions of the physical quantities to validate requirements through simulation. Therefore, the validation phase includes the creation and simulation of behaviour models. The creation and simulation of an expected behaviour model are used for requirement validation in the functional analysis stage. Dimension networks are modelled as a generic behaviour and are executed as concurrent processes.

Although SysML itself is not intended to execute these models, the constraints, functions and associated dynamic behavioural models can be passed to other engineering analysis tools for performing such computation. This approach ensures that requirements and engineering analysis computations are performed on the same system design/architectural model. The RA model, represented in SysML, serves as an integrating framework for other models and development artefacts (Fig. 8). This creates modelling and simulation interoperability, which is implied through logical connectivity between the various design tools and repositories to support collaborative design. The logical connectivity is created using existing protocols (i.e.

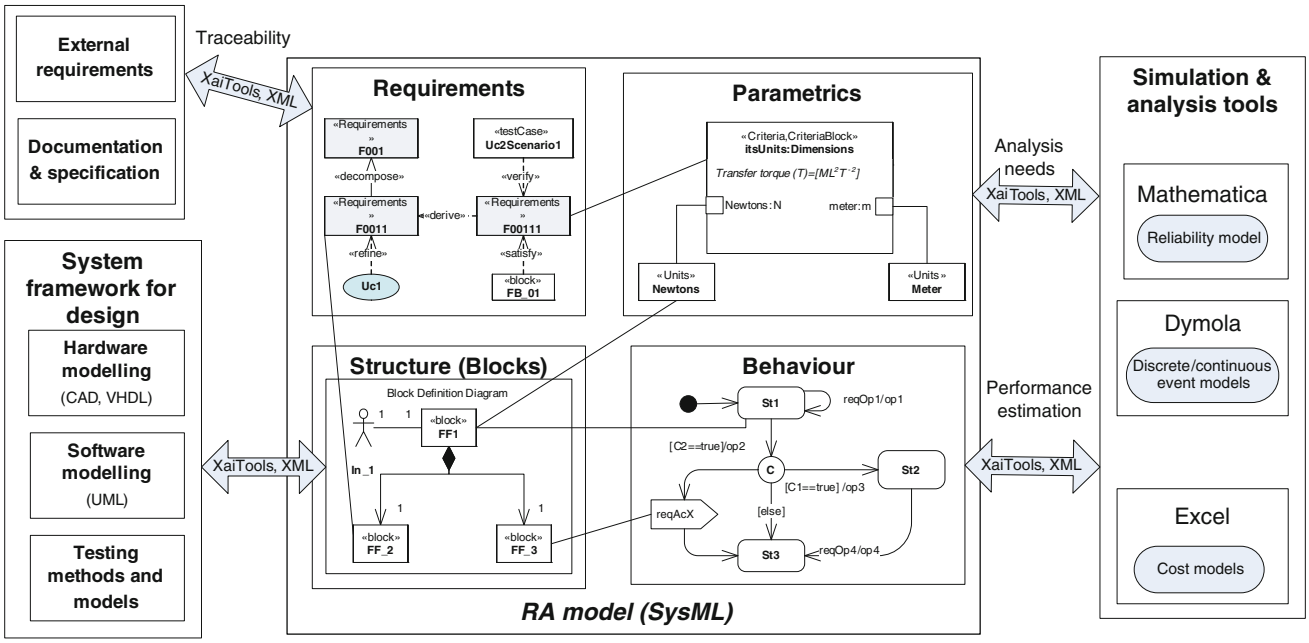

Fig. 8 The SysML RA model as a framework for design analysis and traceability, after [26]

XML, XaiTools). The interoperability is not considered in this work as extensive work is carried out in this direction in the SE community [45]. In the next section, we explain the Workflow SysML diagrams used based on the project application example.

5 SysML diagram usage in modelling process

In the context of this project, the design problem is received in the form of a short narrative statement as follows:

> The task is to redesign a forklift machine to distribute palette goods in warehouses. The aim is to reduce the energy consumption and increase machine manoeuvrability. There is a global market area for the machine. The product should be ready in 2 years. The product cost should not exceed 20,000 euro. The developed machine should be easy to maintain and safety issues should be considered.

This is used to illustrate and explain the meaning of the various diagram types in the CORAMOD. The initial requirement statement is modelled with CORAMOD diagram types in SysML and therefore, to evaluate the approach.

5.1 Specification drivers

A multidisciplinary approach is crucial to analyse the requirements of a complex product. Different fundamental views from the success-critical stakeholders of the system exist during the RA process. Negotiations are necessary to resolve such conflicts and make the essential trade-off decisions. Therefore, support for detecting dissimilarities and inconsistencies between these views must be offered. The requirement negotiation process includes establishing relative priorities based on stakeholder views, identifying issues and conflicts, proposing solution options and making decisions.

The *specification driver* is a support tool and offers a platform for negotiation and setting the context about the requirements and design at an early stage. SysML offers the concept of view and viewpoint to facilitate the modelling of a specification driver. A viewpoint describes a perspective of interest to a particular stakeholder. A view is a type of package that conforms to a viewpoint. The package containment hierarchy provides the fundamental organization of the specification drivers for a particular stakeholder.

The specification driver model starts with a perspective of interest to a set of stakeholders. This forms the viewpoint. In our case study, we consider as an example, the viewpoints of the marketing and engineering design groups. *Profit* is the viewpoint to the marketing group that leads to the question, "Which important factors or *views* will affect profitability during the operations of the system?" For instance, more pallet goods transferred will increase profit (Fig. 9). This means the system should have more power, which generates more energy lost in the current system. The engineering group considers *transfer* of goods as the most important. The views have a dependency relationship by increasing or decreasing the

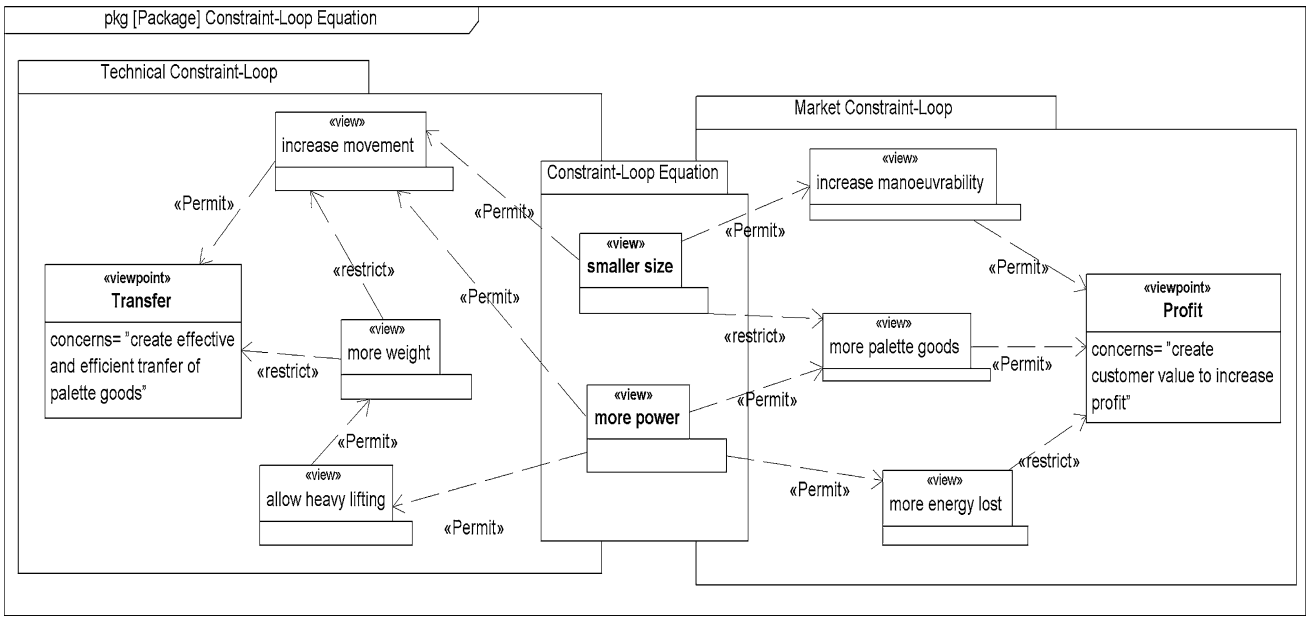

Fig. 9 Specification driver model with viewpoints and views that conform to them

likelihood of a viewpoint. The relationships are modelled as 'permit' and 'restrict' dependency stereotypes. The viewpoints modelled by the various stakeholders can be combined by the requirement engineer to obtain a lower-level view termed "constraint loop equation," with a view shared by all stakeholders. In our case, the marketing and engineering design groups found a *smaller-sized system* with *more power* as a common view to be considered in decision-making and trade-offs. This is also interpreted as primary requirements as "*The forklift machine shall be smaller in size*" and "*The forklift truck shall have more power.*" Some level of importance is clear from the need statement. However, the specification driver is used for providing the analysis to understand the design problem and to come to an agreement between the various stakeholders.

5.2 Decomposition of design needs statement

The customer requirement statement is classified into simple sentences and modelled as initial requirements. From the narrative statement, eight requirements are derived (Fig. 10). These are further prioritized into primary and secondary requirements to indicate the level of importance.

SysML containment notation is used to indicate the relationship between the need statement and initial requirement diagrams as shown in Fig. 10. This signifies that there is no addition or subtraction from the need statement at this point. The decomposition terminates by deriving identified primary and secondary requirements under the checklist stereotypes (Fig. 11). Dependencies among some candidate requirements are identified at this initial stage and the requirements can be traced to the need statement. Areas that need more consideration are brought to view to derive additional requirements. Furthermore, experts allocated to the stereotypes help in directing the requirement engineer to relevant and responsible engineers.

5.3 Checklist functional-performance analysis

The purpose in the functional RA phase is to analyse requirements under the functional-performance stereotype to derive additional requirements and to transform into a coherent description of system functionality (i.e. create system function structure). The steps involved are described in the subsections below.

5.3.1 Black-box analysis

The starting point for this analysis is to concentrate on the overall function of the system. The most basic way we express this is to represent the forklift system as a simple black-box with the conversion of certain 'inputs' to desire 'outputs'. From Fig. 11, the goal or primary requirement (CF0.0) is stated as: "*the forklift truck shall distribute palette goods in warehouses*"

Therefore, the highest level function is for the forklift system to transfer input palette goods to output distributed palette goods as shown in Fig. 12. In the black-box analysis, we concentrate on the external behaviour analysis as the internal behaviour in the black-box is invincible.

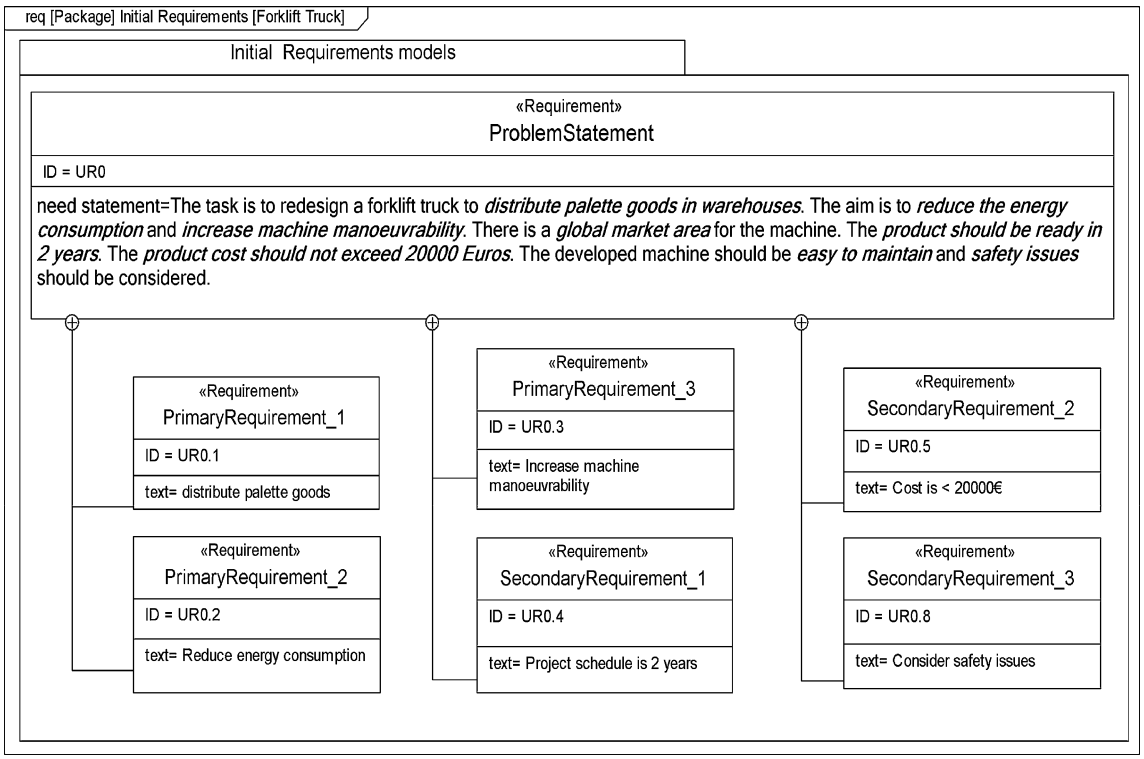

Fig. 10 Forklift truck initial requirement diagram derived from the ultimate requirement

Inside the black-box, there must be a process that facilitates the distribution of the palette goods in the warehouse. For the system to facilitate the distribution of goods in a warehouse there should be some form of control. Studies and analysis of existing product information and conventional system process from the checklist data-management repository (Fig. 7) indicate such a system can be manually or automatically controlled. From the specification driver analysis, manual control is plausible. The black-box use case model is defined (Fig. 13). The key diagram elements are the use case 'distribute palette good' and the actor in this case, 'an operator'.

Since there is active human involvement (i.e., operator), issues like safety, ergonomics and operational cycle should be considered. Some of these influencing factors are identified from other checklist stereotypes. Therefore, this modelling approach creates links and dependencies between requirements in the various checklists stereotypes. A text-based use case description is used to provide additional information to support the use case definition. Each step in the use case description is treated as a requirement and to derived addition requirements as shown in Table 3.

There is no golden rule with regard to the number of use cases needed to describe a system. Use cases can be structured hierarchically and a set of use cases is grouped in an UC-diagram. The use case analysis is further elaborated with a detailed description of its behaviour, using use case scenarios (UCS). Interaction between the actor and the forklift system and the resulting behaviour at the recipient is detailed by the UCS (Fig. 14a). In SysML, the scenario is graphically represented in a sequence diagram (SD). Lifelines in the black-box SD are the 'operator' and the system function 'distribute goods' (Fig. 14a). The UCS allows for deriving the requirements (CF0.05, CF0.06) for the test-case descriptors, the pre- and post-conditions of the use case (Table 3).

The respective functional flow from the UCS is modelled as continuous type behaviour. The behaviour is modelled in a logical sequence with activity diagram (Fig. 14b). The diagram shows the actions required of the operator and the forklift truck to distribute goods. Each action block corresponds to an operation allocated from the sequence diagram. The diagram plays an essential role in the subsequent phase.

The last step in the black-box analysis is the modelling of expected behaviour for the initial validation of the models and therefore, the derived requirements. We derive use case related state-based behaviour from the system operation diagrams (Fig. 14) to visualize and evaluate the proposed models and to make changes. The SysML artefact used is the state machine diagram (StM) (Fig. 15). A StM is to visualize system states and modes and their changes as a response to an external stimulus. It can specify the life-cycle behaviour of a block in terms

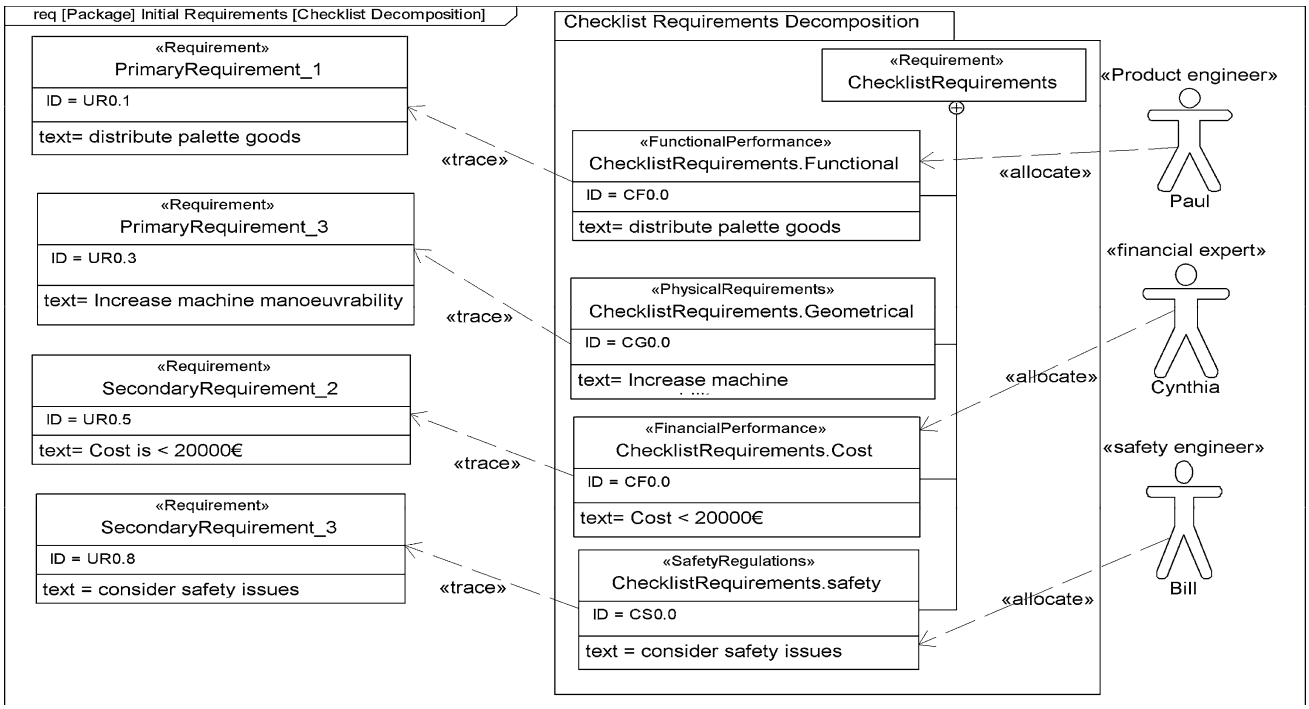

Fig. 11 Importing initial requirements to checklist requirements stereotypes

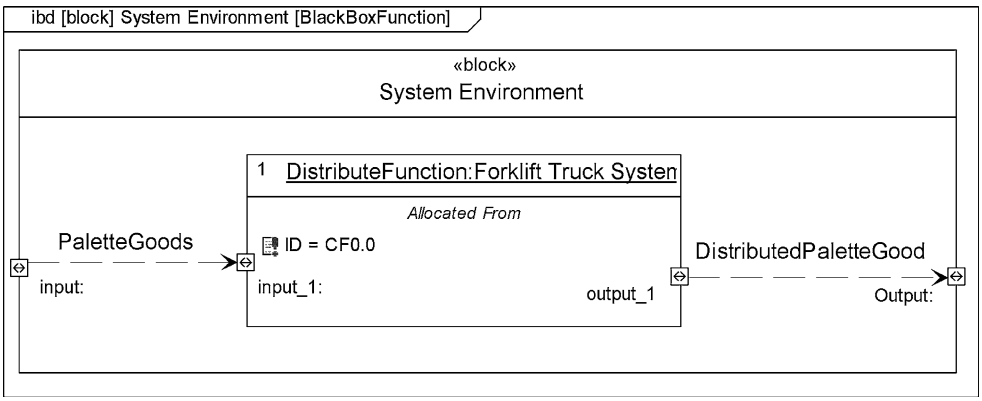

Fig. 12 *Black-box* model of Forklift truck system adapted from [12]

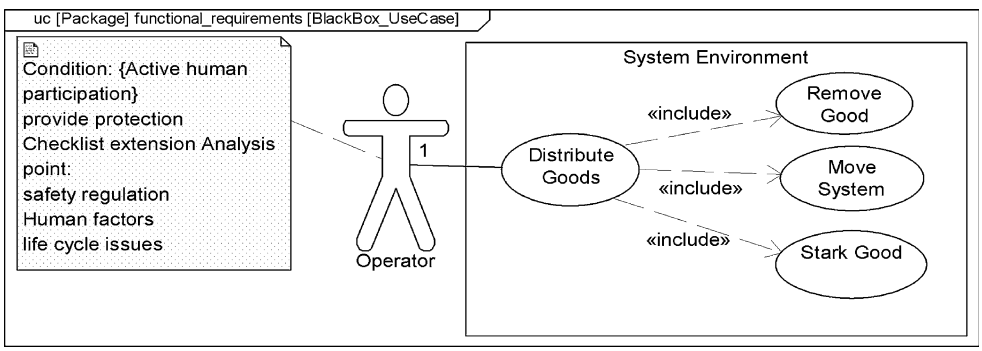

Fig. 13 Forklift system *black-box* use case diagram

of its states and transitions. Validation is performed through model execution using the use case scenario as the basis for respective stimuli. If the proposed scenario is accepted by all stakeholders, the models are imported into the requirement repository for use in the subsequent analysis.

Table 3 Derived requirements with use case descriptions

Use case description	Explanation	Requirement	Id
Pre-conditions	Conditions that must hold for use case to begin	The operator shall inspect the system before start	CF0.05
Other information	To further elaborate actor and subject integration	The forklift system shall be controlled by a human operator	CF0.01
Primary flow	Most frequent scenario of use case(s)	The forklift system shall move palette good in warehouse	CF0.02
Alternative/exception flow	Less frequent scenarios	The forklift system shall remove pallette good from shelf	CF0.03
		The forklift system shall stuck palette good on shelf	CF0.04
Post-conditions	Conditions that must hold once the use case has completed	The operator shall stop the forklift system after completion of work	CF0.06

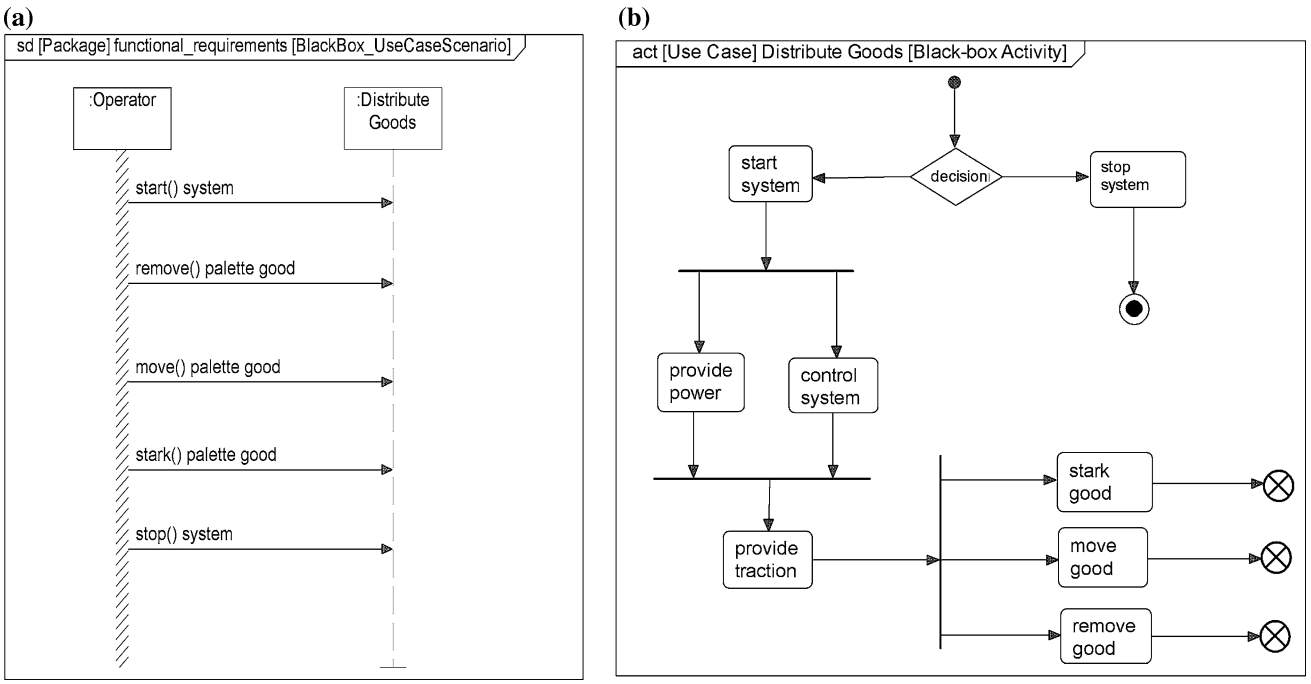

Fig. 14 **a** *Black-box* use case scenario and **b** activity diagram

5.3.2 Transparent-box analysis

The transparent-box (T-box) analysis starts with partitioning and translating information captured in subsequent black-box modelling as context diagram. Based on the use case scenario (Fig. 14), the black-box use case diagram (Fig. 13) is divided into logical sub-functions. The *Forklift system context diagram* is shown in Fig. 16.

The diagram also shows interfaces between key 'input' 'output' elements (i.e., Palette goods, warehouse environment and external entity) in the *System environment*. This modelling approach creates dependencies between requirements in the various checklists stereotypes as some interface entities are identified from other checklist stereotypes. More requirements are therefore derived through the analysis of the identified requirement checklist stereotypes. Furthermore, links are established between the requirements (Fig. 17). The context diagram is defined with the internal block diagram (IBD) to show how the inputs and outputs are connected to the main system. It represents the internal structure of a higher-level block, in this case the system environment block.

The next step is to create function modules. Function modularity is a concept applied to create manageable function models from a larger and more complex function structure. The concept is to extend sequential and parallel process task dependencies to the functions and flows of a function model [6, 15]. In SysML, first, a T-box sequence diagram is defined to identify function flow information (Fig. 18).

T-box sequence diagram is an extension of the previously captured black-box sequence diagram (interaction model) and is utilized to identify the interfaces between the sub-function lifelines. The lifelines are based on the

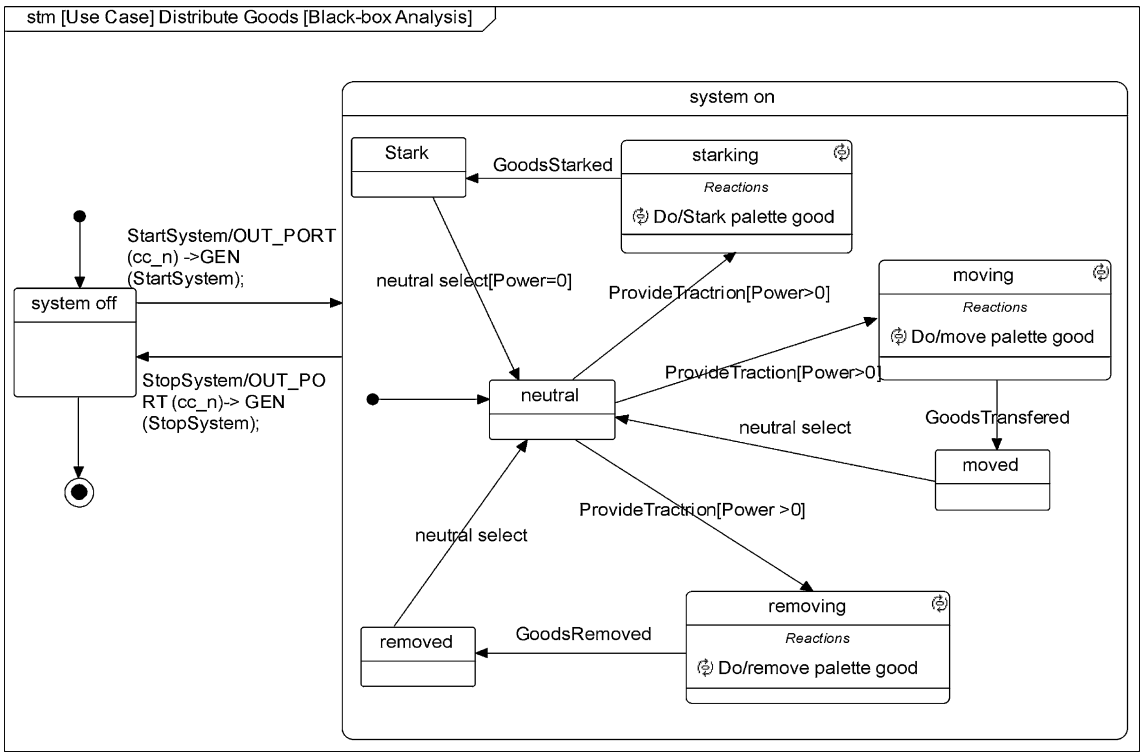

Fig. 15 Forklift system expected behaviour with state machine diagram

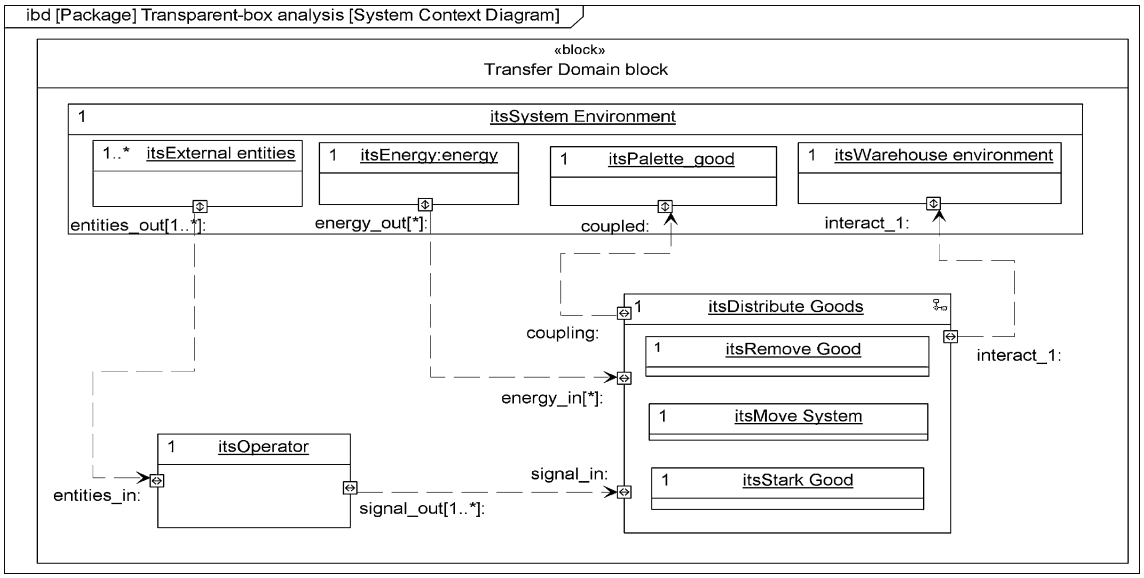

Fig. 16 Definition of transparent-box use case block diagram

identified logical sub-functions in Fig. 16. Data from checklist data-management repository and specification driver analysis are used to help in expanding the use case scenarios.

Once the essential scenario is defined, the next step is to use the sub-function flow interfaces to create the function modules using an activity diagram (T-box activity diagram). The activity diagram is divided into columns or swim lanes along the sub-function lifelines (Fig. 18). Each column corresponds to a function module. The previously defined system operation in the black-box activity diagram (Fig. 14b) is allocated to respective function modules. The function flow is further decomposed using the expanded use case scenario (Fig. 18) as a guideline.

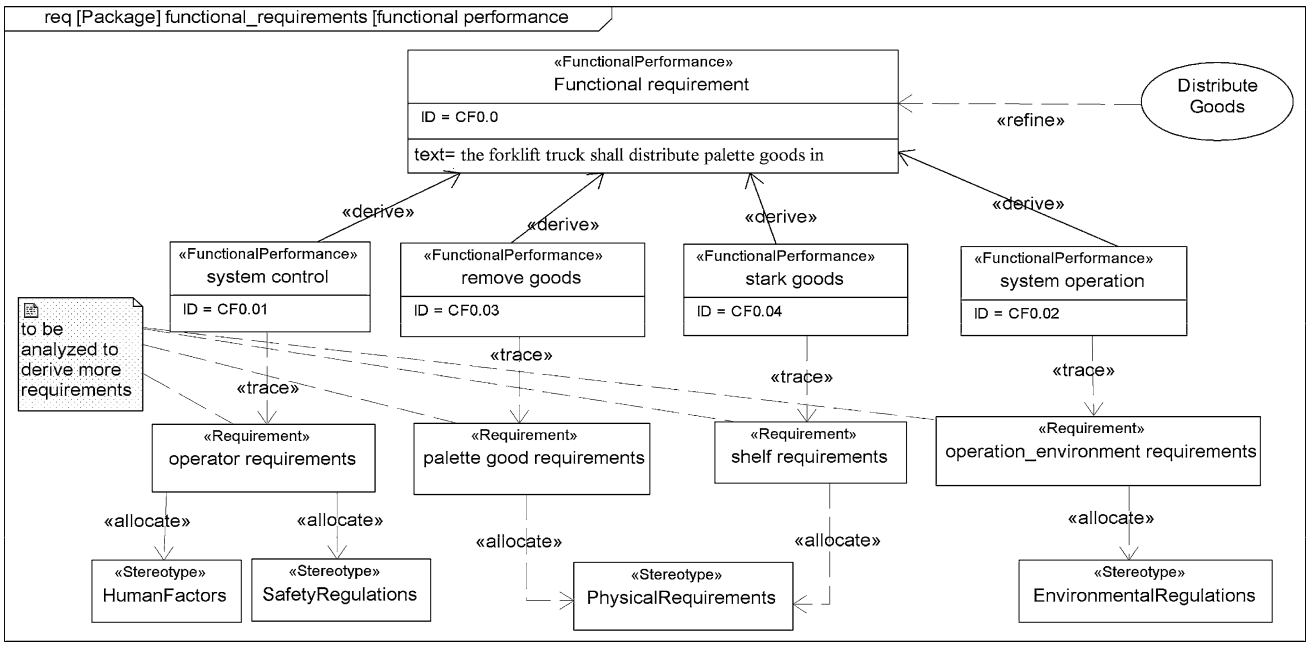

Fig. 17 Derived function requirements

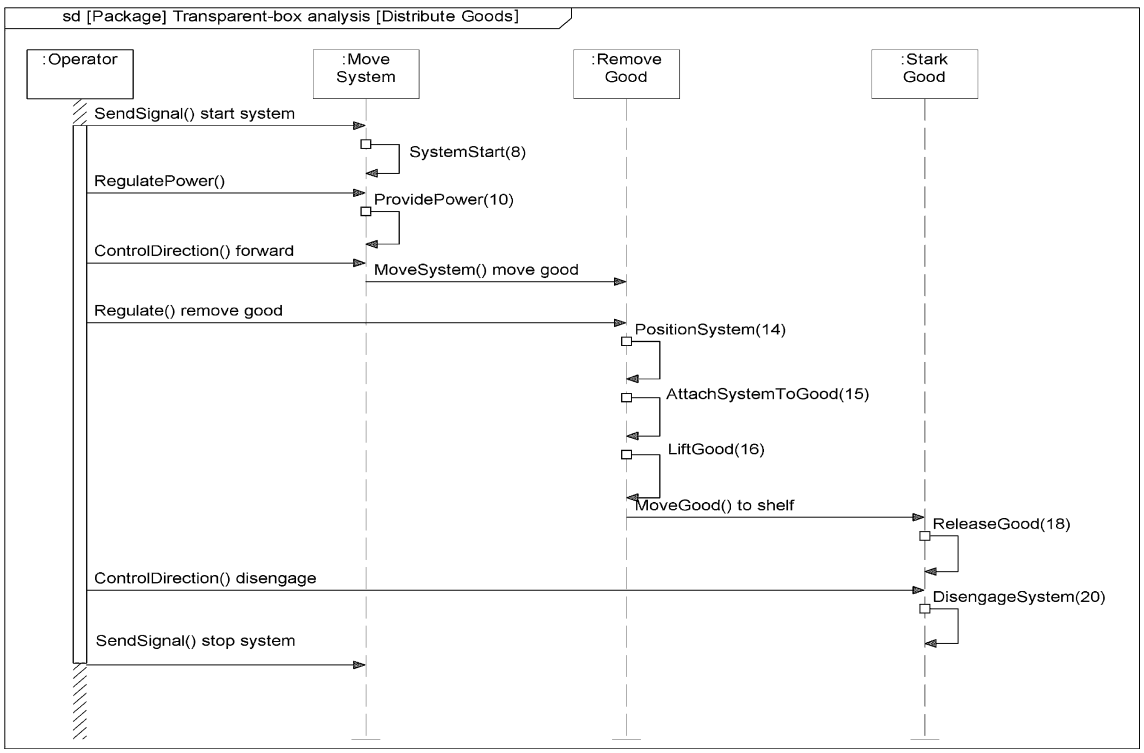

Fig. 18 Transparent-box use case scenario based on decomposition of *black-box* use case

The initial links in the black-box activity diagram are maintained as an essential precondition for mapping. An example of the function module of the forklift truck is shown in (Fig. 19). By extending the activity diagram, important functions may be expanded into separate sub-functions, which identify the input parameters (i.e. the flow of energy, material or information) and the output response of each function. Therefore, the overall system function (function structure) is created by importing the function modules into a common framework. An internal block

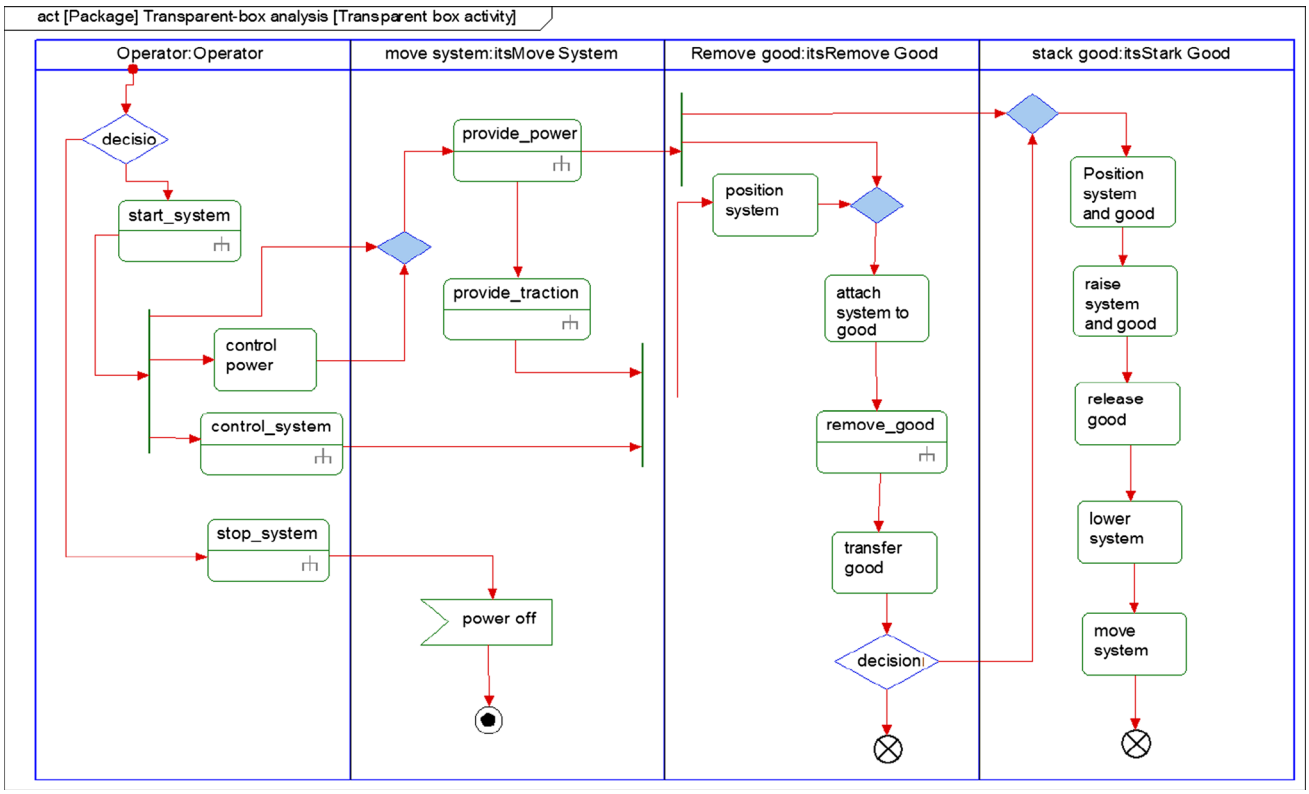

Fig. 19 Transparent-box activity diagram

diagram (not shown) is used to create the system function structure.

The function decomposition comprehensively highlights hidden requirements and forces the designer to consider answers to questions. More requirements are therefore derived and more links are established between the requirements (Fig. 17). Through the systematic decomposition and analysis, a deeper understanding about the design problem from the users' perspective is achieved. The correctness and completeness of the function modules are checked through model execution using state machine diagrams to establish a validation of the derived requirement. Models generated are imported into the data repository for reuse in subsequent phases.

5.4 Checklist non-functional analysis phase

The focus of the non-functional analysis phase is to derive requirements from unbounded requirement statements. In this case, the requirements are first modelled to identify dependencies and to help import information for further analysis. In SysML, an identified non-functional requirement is modelled as an activity hierarchy represented on block definition diagrams.

We consider the operator requirement which is allocated to a non-functional checklist stereotype (e.g. human factor and safety regulation) (Fig. 17). Through the critical analysis of the activities of the operator involving the system in the operating environment, more checklist stereotypes are considered (Fig. 20). Each checklist stereotype is used to critically reason about the operator to define possible requirements. For instance, interaction of the operator and the system is further analysed to derive requirements as shown in Fig. 21. This form of analysis shows the requirement engineer placing himself as the operator operating the system and models the activities involved. In this way, a comprehensive insight and understanding about the operational process is achieved.

The non-functional analysis stereotypes also help to create links with external requirement analysis tools. Requirements under the physical and manufacturing requirements stereotypes can be linked to CAD tools. Traditional CAD tools are used to define geometric (physical form) and the manufacturing description of products. However, some systems incorporate requirement development to analyse requirements from geometrical and manufacturing viewpoints [38, 39]. For instance, the shelf requirement that is allocated to the physical checklist stereotype in Fig. 17 can be linked to a CAGM system to derive additional requirements or import them (Fig. 22). The 'satisfy' connections represent that the shelf geometrical drawing and simulation in CAGM are intended to satisfy each of the three resulting requirements.

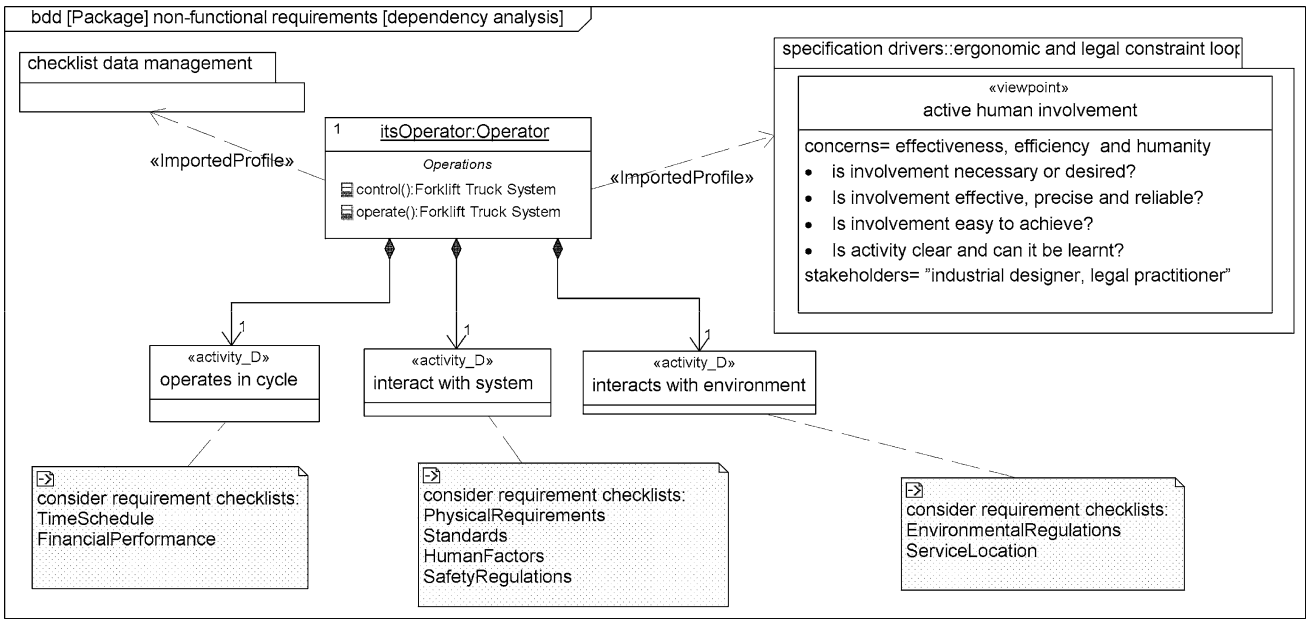

Fig. 20 Non-functional requirement decomposition with an activity hierarchy in a block definition diagram

5.5 Measurable criteria analysis

So far, the requirements identified are in a qualitative format. The next step is to quantify wherever possible for the requirements to be measurable. This is by adding measurable criteria in the form of *value* (e.g. 70) and *units* (e.g. mm) through a *relation statement* (e.g. $\geq$, less or equal to) to the *generic attribute* (e.g. with measurable properties known as dimensions such as distance, length) of the requirement statement [10].

Several generic attributes may be necessary to completely reflect a single requirement statement. Some attributes may also appear in multiple requirement statements. Moreover, several generic attributes may be proposed by engineers for a single requirement statement. In its natural form, requirements act as constraints on the design engineer. The attributes add additional constraints, and not all are relevant. Therefore, there should be a further analysis to seek to define the minimum necessary set of attributes. For that reason, first the attributes are connected together to form a network (criteria network). This helps to manage the criteria relationships and identify criterion, which appears in multiple requirements. Second, the attribute dimensions are modelled to create a network to export for further analysis to validate and reason about the quantified requirements. The following sub-sections illustrate modelling in SysML.

5.5.1 Criteria network development

In SysML, the criteria network is modelled as the non-composite relationship using reference properties (RP). RP can be used to describe a logical hierarchy that references blocks of another composition hierarchy. We take the following derived requirement as an example (CF0.04), '*the forklift system shall place palette goods on a shelf*'. Three requirement attributes are identified as (1) forklift system, (2) palette good and (3) shelf. These attributes in turn have many controlled attributes. For instance, the attribute palette good has size and weight as controlled attributes. *Target values* and *units* indicating tolerance range are then assigned via the *relation entity* and modelled as constraint parameters (Fig. 23). Good judgment is required in value assignment. This is done based on information acquired through some rational process (i.e. market survey, product benchmark, and existing specification) imported from the checklist data-management repository. The network leads to the identification of several requirements in a quantified format (Fig. 24).

After the analysis, a well-formed design specification is established. The requirements obtained so far are now written in specification standard. Good specification standards require that each requirement statement in the specification satisfies several characteristics, including the appropriate use of shall, will, and other keywords. They must also have proper grammar and rigid compliance with a format (i.e. company format). An example of the design specification identified from the criteria network analysis is as shown in Fig. 24.

5.5.2 Dimension network and requirement validation

Since the requirements analysis is a multidisciplinary process, models can be built by different stakeholders.

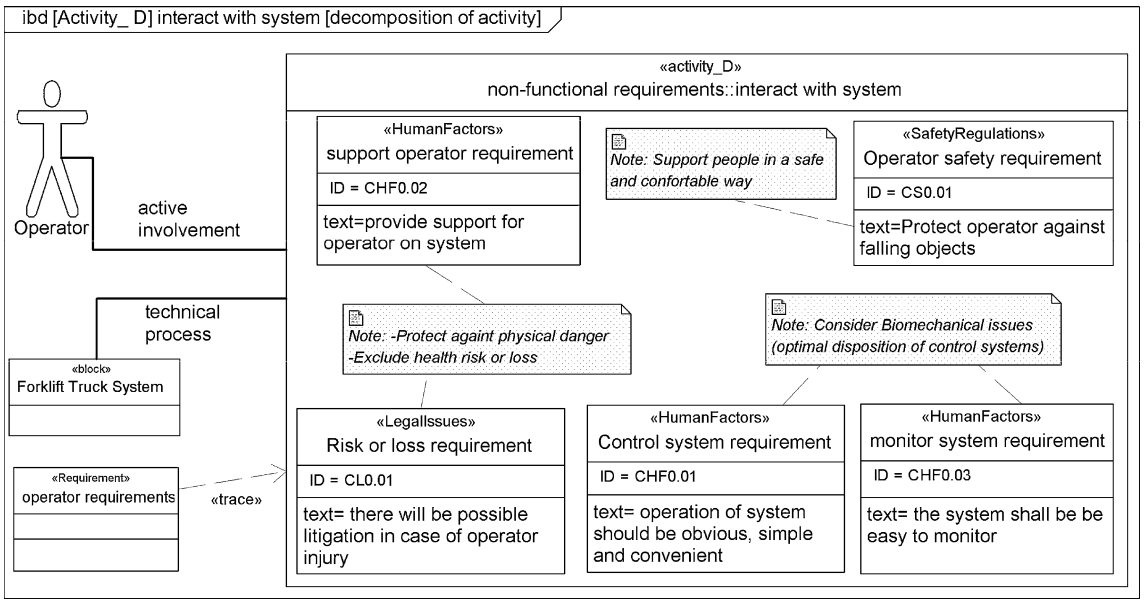

Fig. 21 Analysis of activity blocks identified in non-functional requirement decomposition

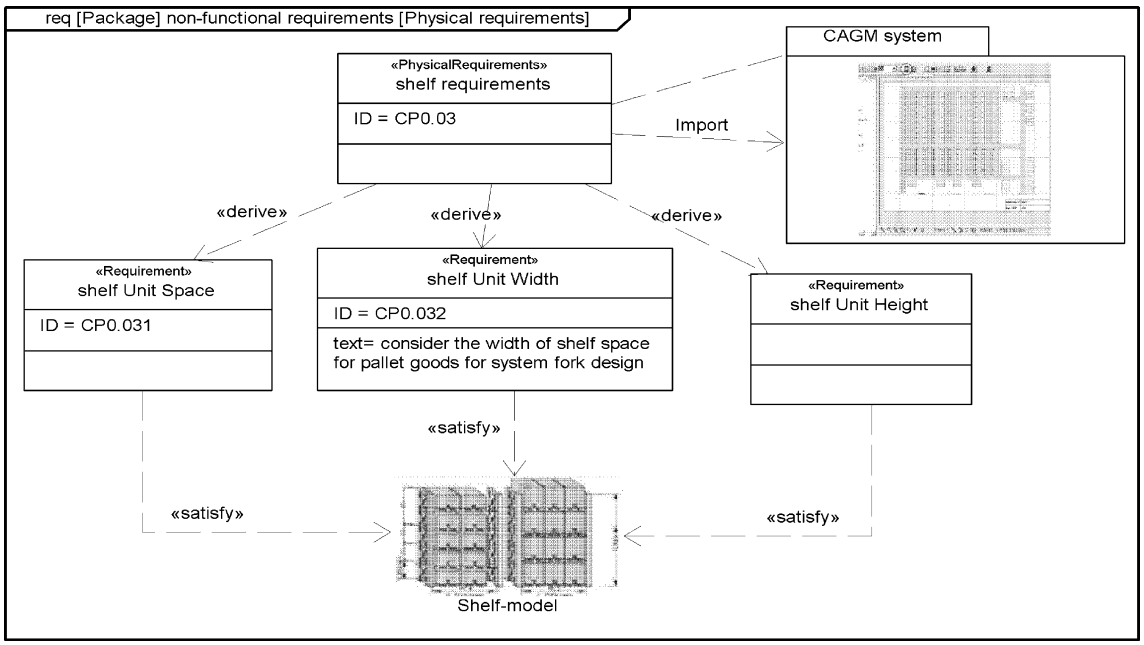

Fig. 22 Non-functional requirement analysis showing link with an external tool

Therefore, a consistency analysis check or validation is done through simulation of the behavioural models. The state-based diagram model execution (e.g. Fig. 15) used to model the expected behaviour to validate the requirements is not adequate. SysML activities are based on token-flow semantics related to Petri-Nets [46, 47]. However, the state diagrams cannot be used to model problems characterized by concurrent action as only one transition is permitted at a time.

Nevertheless, SysML parametric serves as a basis for building system models, which can be exported to external analysis tools for simulation. Parametric integrate detailed design analysis models with the requirements and models for behaviour and structure [45]. The use of the parametric approach in the RA model is unique as the parametric can be used in systematic ways throughout the life cycle to evolve the analysis of design, by starting with the high-level analysis of requirements and evolving the analysis

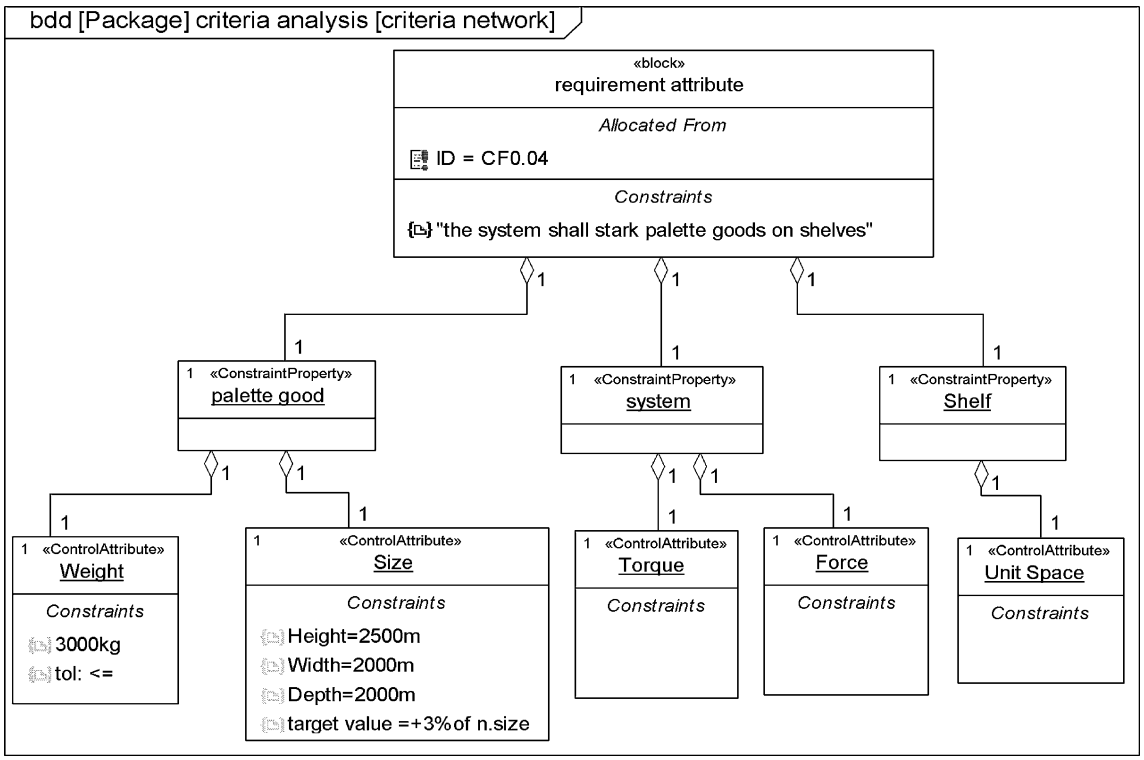

Fig. 23 Non-composite relationship of primitive requirement

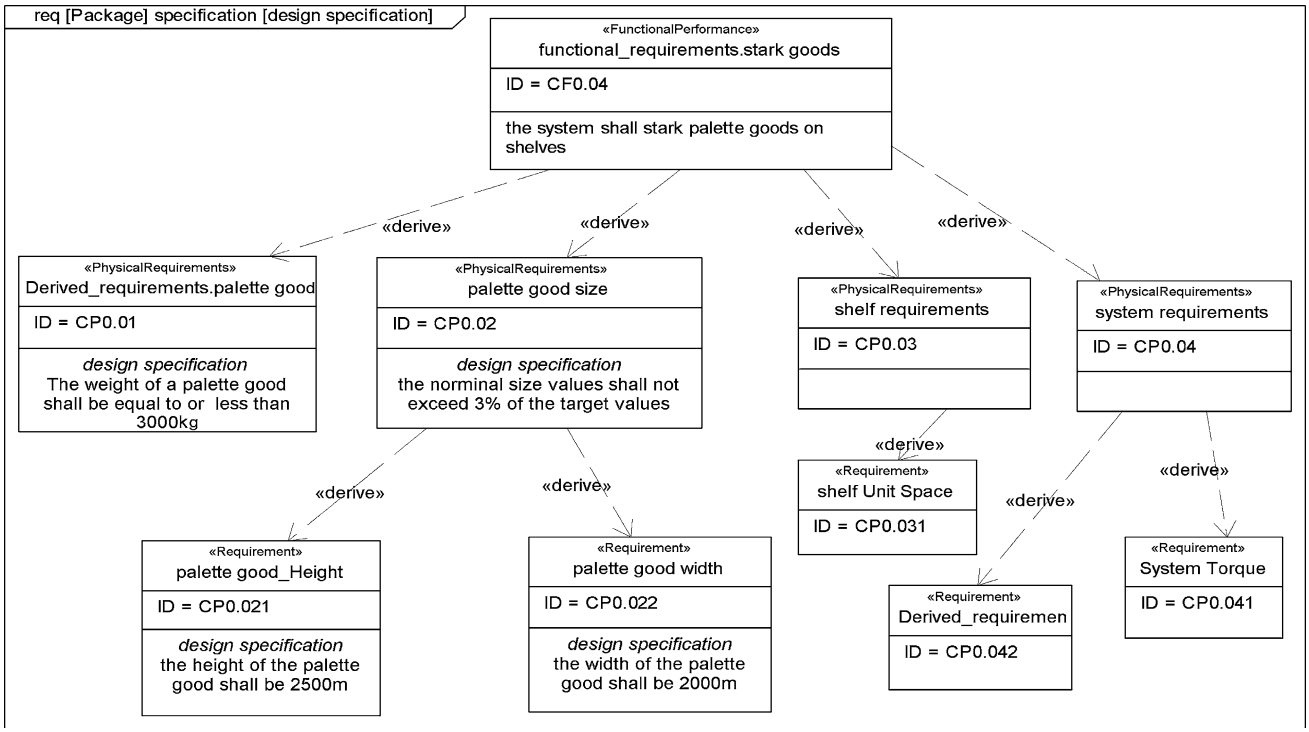

Fig. 24 Abstract of derived design specification in SysML

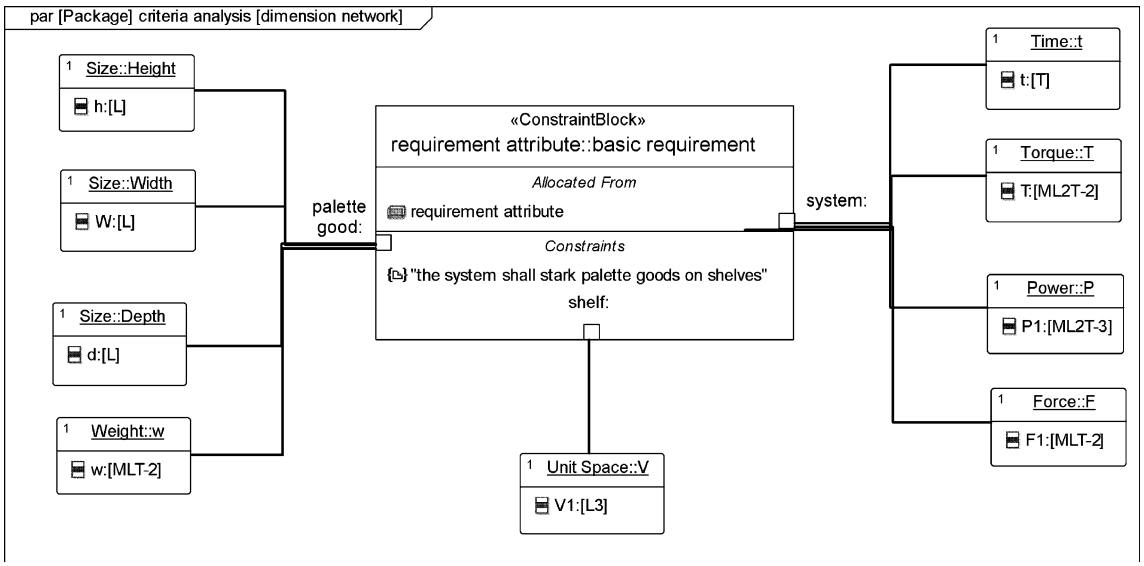

Fig. 25 Parametric diagram showing physical quantities with dimensions

model to include progressively more detail properties of the system.

In this instance, the dimensions of the generic attributes are modelled as dimension networks with the parametric diagram. To model, a *constraint property* of the dimensions of the controlled attribute is created and 'typed' with the constraint block (Fig. 25). The dimension network models created can be transferred using the data transfer protocols (Fig. 8). The network models can be simulation as generic behaviour using the dimensional analysis process. Previous work has been carried out by [48] and will not be discussed in this article. However, this approach helps to compute partial differentials for reasoning about the impact of the various generic attributes. This creates a powerful level to focus on appropriate attributes as the relationship and effect of each requirement attribute can be determined. Non-relevant attributes can also be eliminated.

6 Discussion

The research presented in this paper is part of an ongoing research project to reduce energy consumption in mobile work machines. In this regard, a complete design specification is needed. Consequently, all aspects of requirements that will contribute to the efficient use of the system to reduce the use of energy should be considered. A qualitative research approach, more precisely a case study method was adopted for the research. A case study should use as many sources as relevant to the study to enhance validity and reliability [44]. Therefore, one approach apart from literature review used was active participation in the research project.

Nevertheless, due to the qualitative research approach adopted, the validity of the results reported in the paper warrant discussion. Since empirical evaluation requires the implementation of CORAMOD by engineers in a real industrial setting, it is impossible to conclude the evaluation at this point. The research is ongoing and is in phases. In previous work, the checklist-oriented requirement analysis framework has been developed and evaluated to validate. However, the use of the modelling language environment is validated relatively through a theoretical evaluation procedure. Key issues, which define criteria for design specifications to ensure quality in the early phase of the development, are used for the relative validation [49]. The key issues include well-known problems such as ambiguity, lack of easy visualization, impossibility to analyse and simulate, and integration into existing system development environments and the industry.

In the engineering design domain, according to Darlington [13], the customer needs refer to a mixture of concrete and abstract objects. However, the engineering solutions always reside in concrete objects. Therefore, some practitioners (i.e. experience designers) are quick to ignore the abstract objects and work directly from the concrete objects. This is also due to the creative nature of engineering design activities. Consequently, in most computer-aided design applications, requirements are developed from a chosen concept in the solution space ignoring the abstract objects in the problem space. Nevertheless, according to Restrepo and Christiaans [50] engineering design provides models, with the basis that problem analysis precedes the synthesis of solution. The CORAMOD approach analyze the design needs in the problem space to derive requirements and simultaneously create function

structure as it facilitates the discovery of more requirements and creates best function structure for solution search. This eliminates the sharp borderlines in the conceptual design phase and the gap between the problem and solution spaces.

The engineering design approach has increasingly become simulation-based due to increase in complexity of products. Modelling as well as simulation has become indispensable as physical experiment has become tedious and costly. Traditional CAD tools are used to define the manufacturing description of products using graphical language. While CAE tools are used to solve the discretized and symbolic mathematical problems using analytical language. However, there is a distance between these two tools and together, there is a distance from RA tools. Furthermore, these tools are mostly confined to the later stages in design and difficult to explore in the early stages because of the abstract nature and the use of only semantic language in RA in the domain. In addition, the engineering analysis models that are used may vary substantially in fidelity through the product development life cycle from early abstract analysis models to more refined analysis models. The lack of integration between the abstract RA models, CAE analysis models and CAD geometrical models are aggravated as the complexity, diversity, and number of engineering analysis increases.

Therefore, in CORAMOD, graphical and analytical languages are used in addition to the semantic language to formalizing the RA process. The inclusion of the graphical and analytical language helps to integrate these tools through SysML. The inclusion of the analytical language in the RA process allows modelling with parametric in SysML to allow simulation in the early phase. The application of SysML also provides a unifying mechanism to synchronize conceptual design analysis at the early stage and engineering analysis at the latter stages. It further provides a platform to capture required information and aid authoring and visualizing such knowledge and for users to trace the relationship in RA/CAD/CAE models.

However, there are also limitations for the inclusion of the graphical and analytical languages in the RA model. Clearly, the complex maze of modelling notations used to analyse requirements that are relatively simple and easy to state may be too rigid and heavy-weight for some practitioners. Nevertheless, we maintain that this is not a unique problem as a light-weight approach tends to be a document-driven development which is ad hoc and difficult to integrate in a system development environment.

SysML is geared toward incremental description of the conceptual design where there is notorious lack of efficient tool support. However, the description follows the transformation of textual requirement specifications into product functions and expected behaviour and precedes the design of these functions to product structure. The analysis of requirements is therefore marginalized. CORAMOD approach introduces a systematic and conscious requirement analysis approach into SysML, consequently, extending its use. In addition, CORAMOD in SysML provides an intense use of models, which increases the needed level of abstraction to hide unnecessary system complexity. Expressing the requirements in terms of these models successively leads to the elimination of ambiguities.

Furthermore, SysML is a language not a methodology and it is expected that it can be integrated without many problems into the current development process in an organization. In addition, the language is widely known both in the academia and in the industry, and its use has been accepted in the engineering design domain. The current research project started with an attempt to implement SysML in its raw format. However, due to problems encountered, CORAMOD was developed. SysML provides a structure for the team to work effectively on a large model. The process promotes efficient communication and integration and provides consistency that can be exploited by automation. The formal description of the product at an early stage improves the understanding about the product requirements and how they answer the customer needs.

A major step during requirements engineering, as found in, for instance, DOORS consist of breaking down requirements into sub-requirements. CORAMOD uses the concept of checklist decomposition to break down requirements forming a hierarchy that allows both forward and backward tracing of the requirements to the design need. The concept of the hierarchy also permits the reuse of requirements. In this case, a common requirement can be shared by other requirements. The use of the checklist-oriented analysis is also unique. Technical systems do not operate in isolation and are part of a larger system interacting together, and the effects of the overall interrelationship can be carefully considered at an early stage using the checklist.

The checklist is a generic list of major aspects and sources of requirements. A checklist is a type of informational job assistance to reduce failure by compensating for potential limits to the mental memory. Gawande [51] has revealed several examples, for instance, in the medical field to expose the surprising power of the checklist. In design terms, the checklist may be a list of questions to be used for requirement elicitation, or a list of features to be incorporated in the design. They may also be a list of criteria that the final design must meet. It is the simplest kind of rational design method which unlike creative methods encourages a systematic approach to design [12]. The engineering design can also be verified for accuracy and completeness using a checklist.

In the engineering design domain, the checklist is a powerful tool and has been recommended by several textbooks, for example, [4, 8, 9, 11] as a reference for RA. However, it is limited to basic informational use as a remainder or for verification purposes. The idea of checklist is also used to an extent in software requirements engineering. A requirement checklist technique is used to examine a list of common concerns in the requirement specification [22]. It is used for verification to ensure consistency. Verification checklist is different from that of RA checklist. It defines various aspects that are related to the quality of requirements and specification formats to ensure that the final requirements and documentation format meet the standards of the organization or stakeholder.

One of the principal differences between engineering design now and several decades ago was the emergence of many specialty engineering disciplines (i.e. software, industrial, maintenance, electronic, mechanical). This has come about through the explosion in available knowledge. Therefore, everybody has to contribute to developing the requirements for the system in a group work. The specialty engineer work to identify a minimum set of characteristics of the system and the checklist is used to create a platform for this work. In summary, the use of checklist for RA will ensure a conscious requirement decomposition strategy for generating well-formed (i.e. unambiguous, traceable) requirements, which are easily validatable and verifiable through the use of simulation and for instance, verification-checklists.

7 Summary and conclusions

In this paper, we have presented the use of MBSE to address the RA of complex products. This modelling process with an illustrative example shows how to derive the design specification from the customer requirement. It is essential to have properly structured and controlled design specifications that are consistent, understandable and include all stakeholder requirements and concerns. To achieve this important milestone, we exploited the engineering design systematic methodology by using a checklist structuring approach to define the modelling process of the MBSE environment.

Requirements in a textual format are a wonderful vehicle for communication. However, it is not the best way to represent requirements for engineering products. RA modelling that uses a combination of textual, analytical, and graphical languages for deriving requirements is relatively easy to understand and interpret. More importantly, it gives a straightforward way to review for correctness, completeness, and consistency.

The use of a knowledge-rich requirement checklist makes requirement formulation more model-centric. This helps the designer to eliminate personal preferences improving this way his effectiveness and productivity. The checklist-oriented modelling is also useful in improving discussion and collaboration in a design team. Given that the ultimate requirement does not fully reflect the design problem, the checklist approach allows that all relevant requirements are consciously considered. The checklist-based MBSE presented can help mitigate the risks assumed by the product development enterprise. Such risks include, in particular, the risk of a product not creating value for the customer and matching market needs, the risk of not being able to manufacture the product to specifications and the liabilities resulting from product defects.

The potential of CORAMOD as a method to supporting and facilitate an interdisciplinary engineering approach was demonstrated through the use of SysML. The process shows how to analyse the functional requirements, non-functional requirements and measurable criteria to derive requirements. The discussion steps through the system model development building a sequence of requirement diagram views (specification drivers, functional, non-functional, criteria, and dimension networks). Corresponding requirements are derived via a systematic iterative process that tightly coupled the model development with the checklist RA.

In the CORAMOD paradigm, a holistic visual model is built to enable success-critical stakeholders and decision makers involved in the RA to apprehend the whole problem. The graphical artefacts' built on SysML simplify the communications between the stakeholders, requirement engineers and subject matter experts. The approach visualizes the complex product as functional, non-functional, and behavioural models and contributed to successful requirement development.

Deep knowledge of system behaviour helps to gain a deeper understanding of the system process to derive requirements. Some practitioners may argue that this same methodology can be achieved with a paper and pen in a brainstorming section. There are advantages in using this informal ad hoc approach as for instance direct contact and communication among stakeholders. However, there are well-known problems such as ambiguity, lack of easy visualization and impossibility to analyse and to simulate. Advantages of the ad hoc approach are enhanced with the specification driver modelling that creates a visual negotiating platform and at the same time eliminates the associated problems. Since knowledge management is a focus nowadays, CORAMOD allows for all information to be readily saved and reused, and the process can be verified immediately.

CORAMOD was created with the intension to use SysML to analyse the design need and to connect people and information. Therefore, we propose an extension to the basic SysML requirement diagram, based on a checklist decomposition process for RA. Collaboration makes it possible for the RA team to rise above the system complexity enabled by a central negotiation platform, the specification drivers.

SysML is an UML-based language, which is widely known and used, both in the academia and in industry. Therefore, it is expected that it can be added without many problems into the product development process. In addition, it is shown that modelling requirements through SysML diagrams can be useful to explicitly represent the various ways that requirements can be related to each other and with other design and analysis models. An important quality factor when building complex products is to show traceability. Using checklist stereotypes in SysML is useful to represent a structured decomposition and improve traceability.

CORAMOD is advocated for the engineering design domain. However, today's products include an integration of hardware, software, process and people. Model-based architecture is used to overcome the novel challenges presented by software development. The systematic design approach is used to augment the creative methods in engineering design. CORAMOD is based on these two approaches. For instance, identifying functions leads to the logical design and identification of interfaces; this is a useful approach in both domains. Function quantification results in design requirements that are essential in the engineering design domain to simulate prior to logical design. A properly defined quantified requirement provides a permissible solution space within which to pursue a limited number of solutions. They provide a safe framework within which creativity can be pursued.

The checklist model-based description is our first step towards a semi-automated RA process. The approach will exploit semantic web technologies such as ontologies and knowledge management. A first justification of our systematic requirement procedure has been established. Future work will complete this analysis by focusing on the exploitation of machine readable knowledge representation. This will favour the partial automation of the RA process in the engineering design domain.

References

1. Hall T, Beechem S, Rainer A (2002) Requirements problems in twelve software companies: an empirical analysis. IEE Proc Softw 149:153–160. doi:10.1049/ip-sen:20020694
2. Kauppinen M, Savolainen J, Lehtola L, Komssi M, Töhönen H, Davis A (2009) From feature development to customer value creation. In: Proceedings of 17th IEEE international requirements engineering conference (RE'09), pp 275–280
3. Darlington MJ, Culley SJ (2004) A model of factors influencing the design requirement. Des Stud 25(4):329–350. doi:10.1016/j.destud.2003.12.003
4. Pahl G, Beitz W (2007) Engineering design: a systematic approach. In: Wallace K, Blessing L (eds, trans) 3rd edn. Springer, Berlin
5. Blanchard B, Fabrycky W (2006) Systems engineering and analysis. In: Fabrycky W, Mize J (eds) Prentice Hall international series in industrial and systems engineering, 4th edn. Pearson, Prentice Hall, Englewood Cliffs
6. Ulrich KT, Eppinger SD (2001) Product design and development, 2nd edn. McGraw-Hill International, New York
7. Piaszczyk C (2011) Model based systems engineering with Department of Defense Architectural Framework. J Syst Eng Manage 14(3):305–326. doi:10.1002/sys.20180
8. Pugh S (1997) Total design: integrated methods for successful product engineering. Addison-Wesley Longman Ltd, Essex, UK
9. Dieter G (2000) Engineering design: a material and processing approach, 3rd edn. McGraw- Hill, International Editions, New York
10. Grady JO (2006) System requirements analysis. Elsevier Academic Press, Burlington
11. Otto K, Wood K (2001) Product design -techniques in reverse engineering and new product development. Prentice Hall, Upper Saddle River
12. Cross N (2008) Engineering design methods, strategies for product design, 4th edn. Wiley, West Sussex
13. Darlington MJ, Culley SJ (2002) Current research in the engineering design requirement. IMechE, Part B: J Eng Manuf 216(3):375–388. doi:10.1243/0954405021520049
14. Ullman D (1992) The mechanical design process. McGraw-Hill, Inc., New York
15. Hoffmann HP (2005) UML 2.0-based systems engineering using a model-driven development. J Def Softw Eng(CROSSTALK)
16. McKay A, Pennington A, Baxter J (2001) Requirements management: a representation scheme for product specifications. Comput Aided Des 33(7):511–520
17. Daniela E, Damian H, Jonker CM, Treur J, Wijngaards NJ (2005) Integration of behavioural requirements specification within compositional knowledge engineering. Knowl-Based Syst 18(7):353–365. doi:10.1016/j.bbr.2011.03.031
18. Chakrabarti A, Morgenstern S, Knaab H (2004) Identification and application of requirements and their impact on the design process: a protocol study. Res Eng Des 15:22–39. doi:10.1007/s00163-003-0033-5
19. Gilb T (1997) Viewpoints: towards the engineering of requirements. Requir Eng 2(3):165–169. doi:10.1007/BF02802774
20. Davis A, Zowghi D (2006) Good requirements practices are neither necessary nor sufficient. Requir Eng J 11(1):1–3. doi:10.1007/s00766-004-0206-4
21. Loucopoulos P, Karakostas V (1995) System requirements engineering. McGraw-Hill International, New York
22. Kotonya G, Sommerville I (1998) Requirements engineering: process and techniques. John Wiley & Sons, New York
23. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the conference on the future of software engineering. ACM Press, pp 35–46
24. Dorfman M (1997) Requirements engineering. in software requirements engineering. In: Thayer H, Dorfman M (eds) 2. IEEE Computer Society Press, Los Alamitos, California, pp 7–22
25. Jiang L (2005) A framework for the requirements engineering process development. PhD Thesis, University Of Calgary,

Department of Electrical and Computer Engineering, Alberta
26. Friedenthal S, Moore A, Steiner R (2008) A practical guide to SysML. The systems modeling language. Morgan Kaufmann OMG Press, Burlington
27. Finger S, Dixon J (1989) A review of research in mechanical engineering design, part I: descriptive, prescriptive and computer-based models of design processes. Res Eng Des 1(1):51–68. doi:10.1007/BF01580003
28. Estefan JA (2008) Survey of model-based systems engineering (MBSE) methodologies. International Council on Systems Engineering (INCOSE). Technical Document INCOSE-TD-2007-003-02, Revision B
29. Boucher M, Houlihan D (2008) System design: new product development for mechatronics. Publication of Aberdeen Group, Boston
30. Arthurs G (2008) Model-based system engineering elements for deploying an efficient development environment. Publication of Telelogic, an IBM Company
31. Forsberg K, Mooz H (1995) Application of the "Vee" to incremental and evolutionary development. In: Proceedings of the fifth annual international symposium of the national council on systems engineering (INCOSE)
32. Boehm BW (1988) A spiral model of software development and enhancement. IEEE Comput 21:61–72. doi:10.1109/2.59
33. OMG (2008) Systems modelling language V1.1. OMG Available specification
34. Follmer M, Hehenberger P, Punz S, Zeman K (2010) Using SysML in the product development process of mechatronic systems. In: 11th international design conference (DESIGN 2010), pp 1513–1522
35. Amihud H, Kasser J, Weis M (2007) How lessons learned from using QFD led to the evolution of a process for creating quality requirements for complex systems. J Syst Eng Manage 10(1):45–63. doi:10.1002/sys.20065
36. Rehman F, Yan XT (2007) Evaluation of a context knowledge based tool to support decision making in the conceptual design. In: Proceedings of the international conference on engineering design, ICED'07
37. Tseng MM, Jiao J (1998) Computer-aided requirement management for product definition: a methodology and implementation. Concurr Eng Res Appl 6(2):145–160. doi:10.1177/1063293X9800600205
38. Mamat R, Wahab DA, Abdullah S (2009) The integration of CAD and life cycle requirements in automotive seat design. Eur J Sci Res 31(1):148–156
39. Galea A, Borg J, Grech A, Farrugia P (2010) Intelligent life-oriented design solution space selection. In: Proceedings of the 11th international design conference DESIGN 2010 Dubrovnik, Croatia, pp 1295–1304
40. Davis AM, Hickey AM (2002) Requirements researchers: do we practice what we preach? Requir Eng J 7(2):107–111. doi:10.1007/s007660200007
41. SLATE (2004) System Level Automation for Enterprises (SLATE), now part of UGS Teamcenter Requirements. http://www.ugs.com
42. DOORS (2006) Dynamic Object Oriented Requirements System (DOORS). http://www.telelogic.com
43. Booch G, Jacobson I, Rumbaugh J (1998) The unified modeling language user guide. Addison, Wesley
44. Yin R (1994) Case study research: design and methods, 2nd edn. Sage Publishing, Edition Thousand Oaks, CA
45. Peak RS, Burkhart RM, Friedenthal SA, Wilson MW, Bajaj M, Kim I (2007) Simulation-based design using SysML Part 1 and Part 2,. INCOSE international symposium, San Diego, USA
46. Reisig W (1992) A primer in petri net design. Springer-Verlag, New York
47. Wagenhals W, Haider S, Levis A (2003) Synthesizing executable models of object oriented architecture. J Syst Eng Manage 6(4):266–300. doi:10.1002/sys.10049
48. Brace W, Coatanéa E, Kauranne H, Heiska M (2010) Formalization approach to early design synthesis. Int J Des Innov Res, IJODIR 5(2):25–45
49. IEEE-830 (1984) Guide to software requirements specification. ANSI/IEEE Std. 830
50. Restrepo J, Christiaans H (2004). Problem structuring and information access in design. J Des Res 4(2): doi:10.1504/JDR.2004.009842
51. Gawande A (2009) The checklist manifesto: how to get things right. Metropolitan Books, New York