Proceedings of the 19th CIRP Design Conference – Competitive Design, Cranfield University, 30-31 March 2009, pp24

Requirements Models for Collaborative Product Development

C. Stechert[1], H.-J. Franke[1]
[1]Institute for Engineering Design,
Technische Universität Braunschweig, Germany

Abstract
Multidisciplinary product development in collaborative networks is a typical working condition for nowadays engineers. After describing the main deficits and dangers of such development situations a modelling approach using the Systems Modelling Language (SysML) is shown. The approach focuses on the generation of a requirements model as a basis for discussion and analysis of the real project aims. A short and simplified example from the field of parallel robots illustrates the approach.

Keywords:
Requirements management, collaborative networks, design methodology, parallel robots

1 INTRODUCTION

The multidisciplinary development of complex products is often performed in collaborative networks. Thence, the system is decomposed into smaller and manageable subsystems (or subtasks). Ideally, these subsystems can be handled independently. It is a big challenge to keep the subsystems consistent, because boundaries are diffuse and changes can have an effect on several subsystems at the same time. A goal-oriented approach should consider all important system views during the product lifecycle.

Requirements are seen as the core of a successful product development. All steps of the development process and every subsystem should support the initial goals. A requirements model is shown that uses the Systems Modelling Language (SysML) as a basis. SysML is a widely known standard e.g. in software development, electronics, and automation. A SysML model is able to integrate several different model elements (e.g. scenarios, functions structures, mechanical structures, and manufacturing processes). These elements can be visualised and documented. Based on a classification of requirements and relations (both qualitative and quantitative) a semiautomatic analysis is possible that helps detecting goal conflicts and estimating change impact.

The benefit of the method is shown for high dynamic parallel robots. Parallel robots are mechatronic, typically customised products. Every task at each customer is unique. Thus, the need for a domain integrating method that supports modular systems was the main motivation for this work.

2 PRODUCT DEVELOPMENT IN COLLABORATIVE NETWORKS

In this work, a collaborative network is seen as a group of organisationally and locally separated companies or departments that work together on the development of a special product.

One aim of product development in collaborative networks is to use the specific expertise of different enterprises in diverse domains to generate a domain-spanning product [1], e.g. mechatronic products like robots, cars, or airplanes. Especially, small and medium sized enterprises (SME) cannot provide high expertises in many different domains. On the other hand, a small company can be expert and world market leader in a delimited area.

Another aim of product development in collaborative networks is to decrease the time-to-market by parallelization of work packages. A well-known strategy is simultaneous or concurrent engineering. According to [2] simultaneous engineering leads to 30-60% less costs, 30-90% shorter development time, and 30-87% increase of quality.

Along with these advantages, collaborative networks bring some difficulties and dangers. The most concerning aspects of often-cited difficulties and dangers are summarized and presented from the following seven viewpoints.

2.1 Communication between development partners

The more distant the different sites are the higher will be the effort for communication [3] (e.g. travel, communication techniques) and if no clear code of behaviour is defined, there will be efficiency losses [3]. Different spoken languages and different cultural backgrounds fortify the effects [4].

Thence, it is important to define a common language as a basis for effective communication. This is necessary at least at those high levels of abstraction (e.g. requirements clarification), where many different partners are involved.

2.2 Exchange of information

According to Zanker [5] the storage, selection, and processing of information and the management of knowledge are the main weak points. It is not always clear to every project member whether he works with a valid version of the dataset. In addition, different data formats have to be converted with losses or cannot be handled by every project member [3].

Interfaces are badly defined and no rules for information exchange are specified [2, 3]. If rules are set, it often leads to a higher amount of time spent learning special notations or project-specific procedures. Some product specific approaches (e.g. [6]) were made to assist a project-(data-)coordinator to assure a consistent exchange of model data and make it more time-efficient. However, a development environment has often to be adapted to new technologies. This would lead to problems if the possibility for such extensions was not considered during programming.

2.3 Division of labour

A big danger for efficiency losses is a bad clarification of project goals [4]. The goals must be clear to every project member and the completion of every subtask should be a step towards the fulfilment of a goal.

A bad disjunction of subtasks leads to extra work [4, 5]. Moreover, responsibilities for results are not fixed and not controlled [5]. A single member of the project team is not able to overview the whole system, thence is not aware of weak points and cannot imagine the importance for the whole system [3, 5]. Once established it is difficult to change the distribution of subtasks. The organisational effort for coordination of work and maintenance of interfaces is comparably high [5].

2.4 Combination and integration of results

Often there is no continuous and no systematic workflow present, thus the right results are not available at the right moment [5]. Interfaces are often inexpedient in type and number [5]. Thence, “optimal” results of subtasks (according to the first set of requirements) are integrated and form a suboptimal total system.

The documentation of decisions and results is often seen as a bothersome duty. If any, project members document results at the end of a project or in a way that is just clear for the person who created it. If personnel and boundary conditions change, decisions cannot be retraced and in a worst case situation work has to start over.

2.5 Human behaviour and human relations

Communication between human beings is seen as the main factor to avoid errors, but the more the distance the less is the likelihood of communication. New techniques try to virtually reduce this distance (e.g. videoconferences, application sharing). Nevertheless, some project members are inexperienced or too narrow-minded for using methods and new techniques of communication. In addition, design engineers see themselves as creative inventors, which might lead to internal resistance against the use of methods [3, 4].

Besides efficiency losses due to different cultures and structures in the different companies [3, 5], each partner follows different aims and strategies [3, 5]. If partners do not trust each other, this will lead to “inside-the-box-thinking” of employees, i.e. holding back of information [3, 4, 5].

2.6 Use of methods and tools

Especially in multidisciplinary product development for complex products a huge variety of different methods and tools is used [7]. Different companies apply different methods for the same type of problem. Furthermore, experts often use their favourite tools (e.g. the use of a specific CAD-system sometimes seems to be a “philosophical” rather than a technical/economical question).

It follows that project members use methods wrong, not adapted to the actual problem or not at all [4]. If methods are used a broad number and variety of methods are used that cannot be handled easily [4].

Van Beek and Tomiyama [8] state that the integration of different methods is the main challenge. Some approaches (e.g. [6, 9]) already showed that it is possible to create integrated development environments. However, these are limited to the predefined set of implemented methods. Furthermore, special formalisms are used to apply the methods.

Following an idea stated in [10], Figure 1 shows the benefit of formalism as a curve over the degree of formalism. Apparently, the maximum benefit lies in between a too low and a too high degree of formalism. It is clear that the run of the curve highly depends on the type of project. The narrower the boundary conditions (i.e. number of domains, project members involved) the more moves the curve to the left. Thence, Figure 1 shows a useful area of formalism located around the optimum of the described curve. Figure 2 shows a similar approach of [11] now distinguishing between the effort for exchange and standardization of information. A minimum effort (though a maximum benefit) is reached in between recommendations and norms as a measure for standardization grade. However, if the norm exists and is well known, the effort for that standardization will not count for the actual project.

2.7 Organizational aspects

Limited resources, time pressure, dynamic boundary conditions, and goal conflicts affect performance and productivity [4]. Decision making processes and managing of development goals are main weak points [5], i.e. definition of goals and if necessary their adjustment due to changing environments.

Due to hierarchy (e.g. steering committee, project management, and work team), domain (e.g. mechanics, electronics, software) and place (e.g. different companies/ departments, different sites) a number of somehow independent “islands of knowledge” emerge. Each island uses a specific subset of the whole project knowledge and is often not sufficiently connected to others [3].

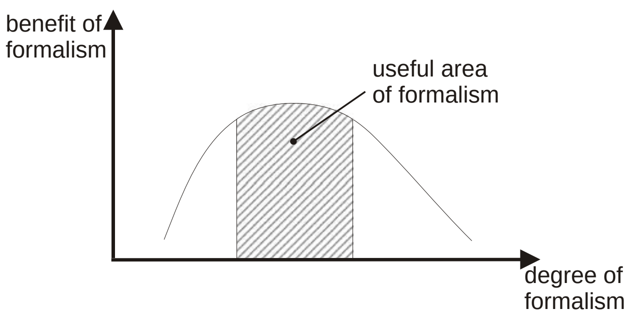

Figure 1: Benefit of formalism, related to [10].

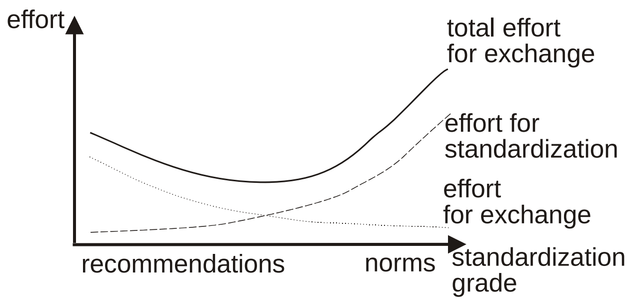

Figure 2: Effort for exchange/porting depending on standardization grade, [11].

The organisation of a collaborative multidisciplinary project for the development of complex products needs a big effort in planning before the project starts. Main points that have to be considered are scheduled time and cost frames. Also the development risk and the possibility of changes (e.g. of customer wishes, laws, available technologies) are main influencing factors for the success of a project. A very comprehensive summary is given in [12]. Here, the key elements of successful operational design coordination are identified as coherence, communication, task management, schedule management, resource management, and real-time support.

3 MODELLING THE PRODUCT

As mentioned earlier the development of a product makes use of a variety of different models or "partialmodels" that describe a specific view on the whole system. Two definitions of model are cited below:

"Models express relations between real conditions in an abstract form. As a copy of reality models own simplifications that on the one hand cause a loss of realness, but on the other hand bring transparency and controllability of real relations." [13]

"Model is a purpose-dependent, finite, simplified, but still adequate representation of whatever is modelled, allowing us to abstract from its unimportant properties and details and to concentrate only on the most specific and most important traits." [11]

Both definitions contain the aspects of abstraction and simplification to bring transparency and to concentrate on the most important characteristics. Additionally, Ort [13] states that models bring controllability of real relations: We cannot control what we do not understand.

3.1 Requirements on models

According to [11] a model should represent the subject to be modelled, ignore unimportant details (abstraction), and allow a pragmatic usage. The purposes are to support and improve the understanding of the matter and build a common basis for discussion and information exchange. Moreover, models should allow comparison of different solutions as well as analysis and prediction of behaviour and characteristics of the system to be designed. The organisation of a model should contain its structure and architecture. Furthermore, interactions between components, component interdependencies, and important external relations should be taken into account.

One important aspect of a model is its representation, especially the visualization of its contents. Salustri et al. [14] mentions "there is relatively little use of diagrammatic visualization of qualitative information in the early stages of designing", although designers are seen as "visual thinkers". Within this context two principles are stated [14]:

- Simplicity is power.
- Diagrams augment cognition.

A model of the designed product is always some kind of documentation, too. It documents the actual state of work, allows for discussions, and gives a basis for presentations to e.g. stakeholders. If personnel changes during the project the documentation should help the new project member to acquaint himself with the topic. For follow-up projects, the rationales for decisions are helpful for the development of a new product or for reconfiguration of the product in operation.

The necessary degree of formalism depends on the project and its state. The more creativity is demanded the more would formalism hinder. For instance, in the early phases a good designer would start with a freehand sketch rather than using a CAD system for his very first ideas. On the other hand, in later phases a formalized workshop drawing or a detailed 3D-CAD model is necessary to exchange the generated information in a commonly understandable language and to transfer information to other models, e.g. FEM. Many product development processes thus suggest a from-rough-to-detailed approach. The model should just be as detailed as necessary to provide a commonly understandable basis for all persons who might get in touch with this model. Wherever more detailed aspects are needed, a submodel for a subset of project members should be generated.

3.2 SysML

The Systems Modelling Language (SysML) is an approach to model a product on different levels of abstraction and with different viewpoints. It is a widely known notation within the fields of software development, electronic design, automation, and (in parts of) mechanical engineering.

SysML uses parts of UML (Unified Modelling Language) and special extension for systems modelling (e.g. requirements diagrams). Commercial and open source modelling tools support UML and SysML profiles. OMG SysML v1.0 was issued as Available Specification in September 2007 [15] and provides a common basis, so a better exchangeability, to describe requirements, as well as structure (e.g. blocks, packages, constraints) and behaviour (e.g. activities, use cases). So far, SysML mainly focuses on the needs of software and electric development, but new profiles containing new classes and stereotypes can be generated for customisation. All relations can be visualised in diagrams and formatted to desired views. Moreover, view-specific lists (e.g. requirements lists) and matrices (e.g. Design Structure Matrix (DSM), traceability tables) can be generated.

Since UML and SysML are widely known and taught at universities, most project members do not have to learn a new and complex modelling language. It can be assumed that this leads to a higher acceptance than for completely new notations. Of course, some new and project-specific elements have to be integrated. In [16] functions structures and in [17] an airplane structure is described with UML class diagrams. This shows that existing methods can be flexibly integrated and handled with the existing tools.

Commercial tools already provide a variety of useful functions. For instance, view specific requirements lists and traceability matrices can be generated from the model and then used e.g. in standard office software. Macros can be programmed to extend the functionalities. Client-server architecture allows the collaborative work with the model at different sites and security procedures are already implemented in the software. The principles of versioning (coming from software development) allow a simultaneous collaborative work on the models.

One drawback of commercial software is the limited access via defined interfaces and the possible change of these interfaces with a new version. Another is the cost of licenses, which is important especially for SME. One possibility to overcome these is open-source software that however brings other disadvantages with them.

3.3 The requirements model

In a collaborative network, as in every major development process the system has to be decomposed into a smaller, manageable, and at the same time consistent set of

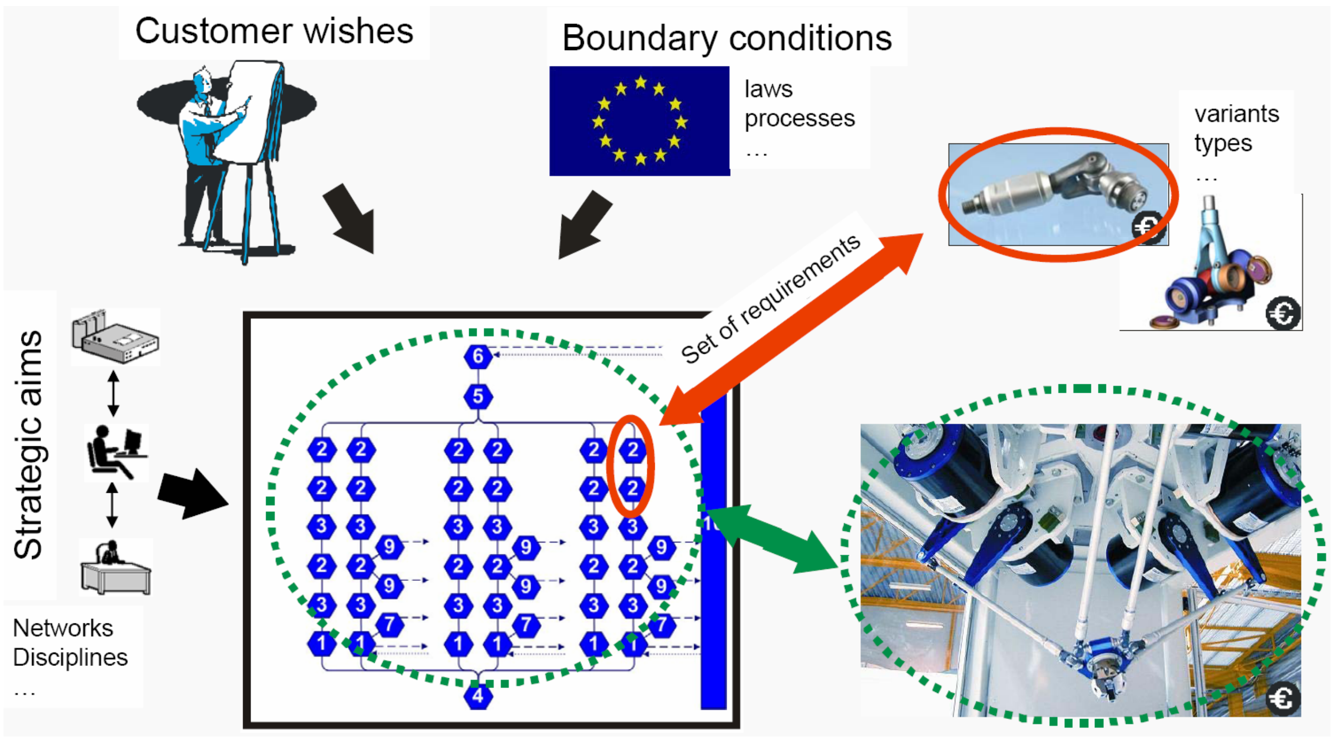

Figure 3: Requirements and constraints for the parallel robot HEXA (schematic overview).

subsystems (subtasks). The requirements model is the core that forces the development process to fulfil the initial customer needs, the companies' strategic aims, and other constraints, e.g. arising from laws and regulations. Figure 3 illustrates this thinking approach as a schematic overview for a parallel robot of type HEXA (six DoF).

The requirements model is one of the first models of the product. However, it is not complete a priori. The further the product development process advances, the more requirements evolve. This is due to the augmentation of knowledge. The better the understanding of the matter is the more detailed requirements are specified and the better will the idea of boundary conditions be. Furthermore, after each made decision (e.g. choosing a solution or design principle) new requirements evolve. For instance, the decision "rack should be welded" leads to requirements like "use steel profiles". The decision "rack should be casted" would lead to very different requirements like "allow for big radii".

Experience points out that for typical projects around 50% of the requirements evolve after the "clarification of task" phase. Figure 4 is taken from [18], who analysed interdisciplinary product development. In the first phases the intensity of a conscious clarification of requirements is high, but decreases to a very low amount within the first third of the project. The documentation of requirements

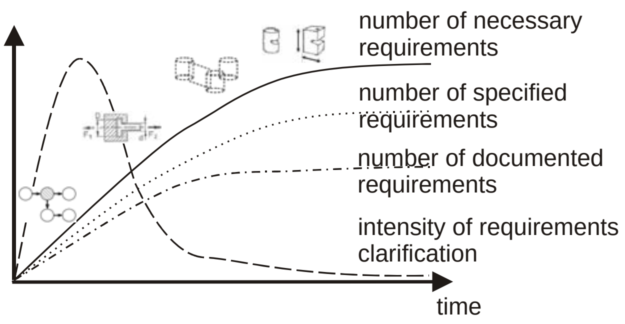

Figure 4: Qualitative display of requirements clarification during a project, according to [18]

does not advance after this phase. Nevertheless, requirements are specified also in later phases, but not documented. However, the number of necessary requirements is higher during all phases. In [19] a holistic approach of requirements management within a PLM context is demanded.

This paper focuses on four aspects concerning the requirements model: Surroundings, structure, relations, and analysis.

Surroundings

One of the first steps in the development process is to analyze the product surroundings in order to recognise the important requirements (e.g. [20]). Here, the whole product lifecycle has to be taken into account including different scenarios (e.g. [21, 22, 23]) or use cases, with all related actors, surrounding environment and possible disturbances. Furthermore, the different product views from different domains have to be considered and those requirements generated by later development steps (e.g. simulation, manufacturing) should be gathered. A systematic documentation helps to identify and to use the collected requirements and constraints. It further shows which requirements derive from what surrounding elements. If several requirements from different domains cause trouble according to the same surrounding elements it might be interesting to think of extending the system boundaries.

Structure

For a better accessibility, information should be structured [24, 25]. Requirements can be structured in a hierarchy such as goal, target, system requirement, and subsystem requirement. Furthermore, they can be allocated to a domain and to a purpose in the development process. Getting more concrete requirements can be allocated to their concerning subsystems. In addition to well-established attributes (wish / minimum / fixed, source) it makes sense to assign certainty and change probability.

Relations

Model elements are related to each other. In earlier work [26] a basic classification for relations was suggested according to development steps, granularity, support, direction, linking, and quantifiability. The systematic integration of relations into the model helps designers understanding the matter and be aware of interfaces to other disciplines. During synthesis steps, getting aware of relations might lead to new model elements. In [18] this was shown for an interdisciplinary development in the field of medical apparatuses. In addition, the modelling of relations is the basis for the analysis of the model, discussed below.

Analysis

One important aspect during the development of complex products is to detect goal conflicts both in early qualitative and later quantitative phases. The earlier one gets aware of possible goal conflicts the higher will be the benefit. This means not just to reject possible solutions early, but also to be aware of problems that might occur in later phases and be prepared for their solution.

Often goal conflicts do not appear on abstract level, but due to decisions on a more concrete level. On these levels, goals are described as (technical) requirements and allocated to the total system or to components. Parameters of some components are already established. As the system is subdivided into many different subsystems of different domains and diffuse boundaries, it is difficult to trace relations without systematic assistance. Hence, the traceability is important to follow the traces from the requirements model through a number of more or less concrete partial models and back.

In addition, the impact a certain change will have on the whole system can be estimated by following these traces. If a boundary condition changes during the development, the analysis shows the effected areas of the product. It thus helps to decide on how and where to adapt the actual concepts or to start all over again.

If in dynamic environments one knows the (un)certainty of a specific requirement it is possible to plan the development process efficiently (e.g. focus on the certain aspects and leave uncertain areas solution independent as long as possible).

According to [10] a system is characterised by a bigger number of relations between system elements within the system boundaries than outside them. The subsystems of a product are to be developed as independently as possible. This approach might lead to modular systems with redundant structures, i.e. synergetic effects are not used and the same problem is solved twice, because module interfaces are generated just because of departmental or domain separation. In addition, the development of modular systems often focuses on just one aspect of the product (e.g. assembly). The requirements model describes early the real aims of the modularity and by analysing the relations modules can be separated more purposefully.

4 PARALLEL ROBOTIC SYSTEMS FOR HANDLING AND ASSEMBLY

Within the Collaborative Research Centre 562 “Robotic Systems for Handling and Assembly – High Dynamic Parallel Structures with Adaptronic Components” concepts for design and modelling of parallel robots for high operating speeds, accelerations and accuracy are developed. Due to the use of closed kinematic chains, parallel robots feature relatively small moved masses (drives are mainly placed in the rack) and high stiffness. In comparison with serial mechanisms, they offer higher dynamics and high accuracy, especially when new and optimized structure components (e.g. adaptive joints [27] and rods [28]) are used. The disadvantages compared to serial robots are mainly a small ratio of workspace to installation area and the existence of singularities within the workspace. Thence, new design, analysis, and control methods were developed to overcome these drawbacks. As a mechatronic product, several disciplines and many different partial models [7] are necessary to set the robot in operation. This results in relatively complex products with complex relations.

As by now, no parallel robots are sold as mass products but customized to the needs of a special customer. The re-use of knowledge, thence the configuration through a modular concept and an effective change management through a systematic holistic view are helpful to provide the desired fast time-to-market as well as high quality and optimal products to the customer needs.

The developed SysML-based requirements model reduces the abovementioned difficulties of collaborative product development providing transparency, communicability, exchangeability, and coherence. The following examples illustrate the approach.

Figure 5 shows in the explorer view the packages of a model for a project “New Robot”. Besides the SysML and UML profiles, one can see the packages “Requirements” and “Surrounding”. The requirements are hierarchically distinguished into “Goals”, “Targets”, and “Technical Requirements”. “Surrounding” documents the “product environments” and “Use Cases”. For instance, one use case in the product lifecycle phase “Use” describes the handling of muffins. This use case leads amongst others to a refinement of the requirements “workspace” and “payload” (see Figure 6). For each customer a unique use case has to be developed, considering the specialties of

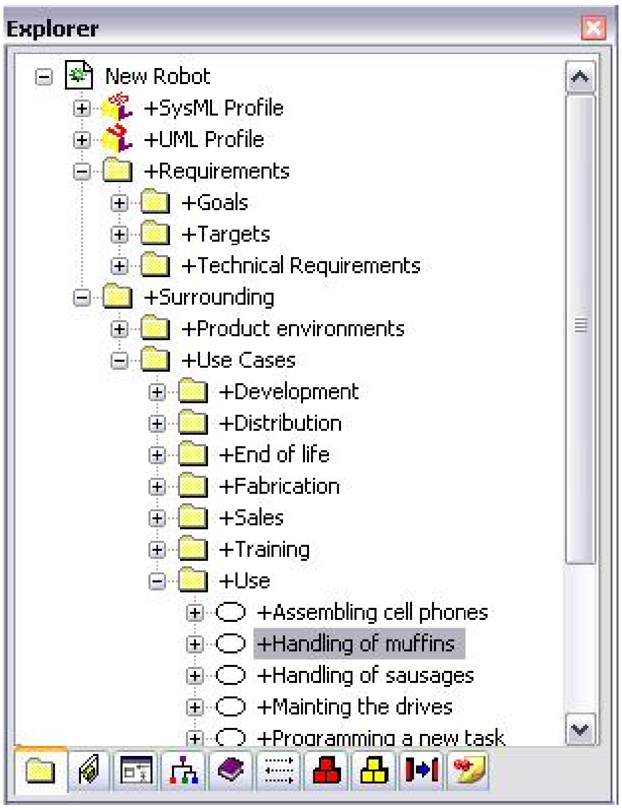
```
Explorer
New Robot
    +SysML Profile
    +UML Profile
    +Requirements
        +Goals
        +Targets
        +Technical Requirements
    +Surrounding
        +Product environments
        +Use Cases
            +Development
            +Distribution
            +End of life
            +Fabrication
            +Sales
            +Training
            +Use
                +Assembling cell phones
                +Handling of muffins
                +Handling of sausages
                +Mainting the drives
                +Programming a new task
```

Figure 5: Requirements and surroundings for a new robot in the hierarchical explorer view.

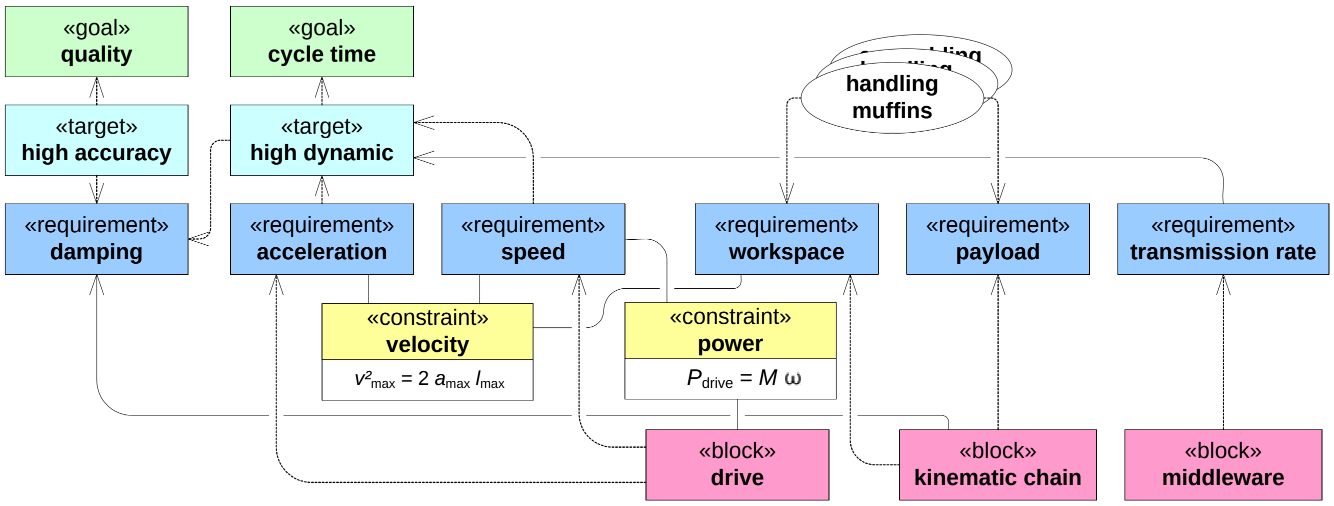

Figure 6: Goal oriented view on the product, excerpt of an extended requirements diagram based on SysML.

that specific surrounding. For instance, the number and arrangement of conveyor belts, the size and weight of the objects, and the following planned manipulation steps have to be considered. It might be that the surroundings show synergetic or parasitic effects. For instance, if a mechanism could help to orient the objects in a way that the robot needs one DoF less, the whole systems could get cheaper, especially regarding Total Cost of Ownership (TCO), i.e. energy costs. However, the new use case shows similarities and differences to already performed projects.

Besides fulfilling the specific use case “Handling of muffins” one important goal is a short cycle time. There are a number of targets that support this goal. However, not every is related to the development of the robot, i.e. not within the projects boundaries. For instance prior or following manipulation steps can decrease the cycle time by supplying the objects in a more efficient way or by picking them in a more flexible way (e.g. objects are

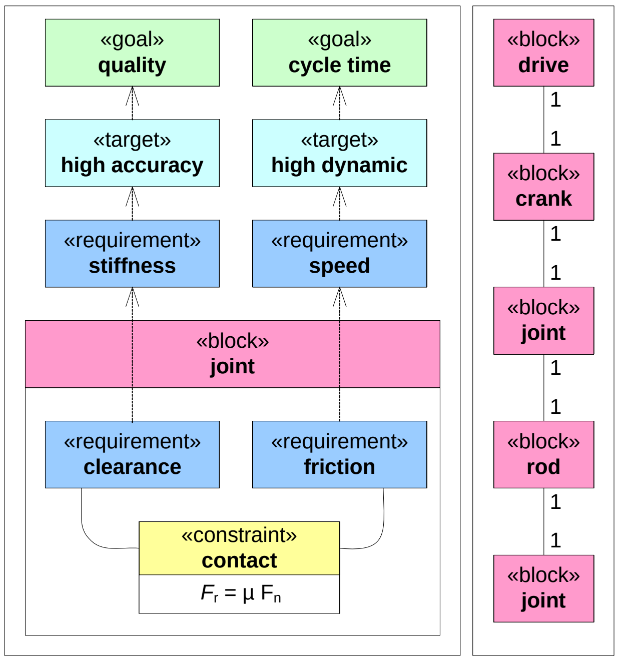

Figure 7: Simplified example of the hierarchical view on requirements of components (left) and component structure in a kinematic chain (right).

grouped relatively to each other, but not absolutely to a fixed coordinate system; another robot picks the grouped objects and orients them). The target “high dynamic” is directly related to the robot and supported by the requirements “high acceleration” and “high speed”. As a simplified example, the triangular relationship between acceleration, speed, and workspace can be described by an equation considering constant acceleration at the tool centre point (TCP). As long as the concretion level is low, this simplified equation can just give an idea of a reasonable area. However, it contains the danger of prejudgement.

Another supporting requirement of “short cycle time” is the transmission rate of the middleware (right side of Figure 6). That means, even if the robot accelerates pretty fast it would not necessarily lead to short cycle times. If it had to stop and wait for new data (because of delay in transmission), the high acceleration would gain nothing. If the acceleration would be designed for the special task in such a way that no waiting periods evolve, drives could be cheaper or less energy consuming. Developers from these different domains should thence work together to find the optimal – goal supporting – solution. The diagrams thus represent a common level of knowledge of the whole system and its relations. Moreover, the overall development goals are always present to each developer. At the high-level representation, a domain spanning discussion is supported. Then in the software development domain a more detailed view on the middleware is necessary. For this purpose developers use e.g. sequence diagrams that are supported by the SysML notation and modelling tools, too.

The combination of targets “high dynamic” and “high accuracy” lead to the requirement “damping” (see left side of Figure 6). The kinematic structure is designed following lightweight principles to fulfil the target “high dynamic”, thus an internal structural damping is relatively small. When the robot stops the structure oscillates, thus the accuracy is affected. According to the lightweight structure, the oscillations die out relatively slowly. Adaptive components are able to suppress oscillations actively [29]. The adaptive components initiate oscillations opposite in phase, thence accelerate the die out. Many different disciplines are now involved. The adaptronic components (e.g. planar piezo actuators) have to be physically integrated on the rods, a suppression strategy must be developed, and control hard- and software must be designed. However, this technology leads to additional costs in development, manufacturing, use, and recycling. The consideration of the product surroundings show that the technology leads to big benefits in the area of precise handling or assembly, namely to gain better cycle times. For some other use cases, the benefit would not justify the costs.

Figure 7 shows on its right side the decomposition of the kinematic chain into its structural components. Regarding the kinematic chain in relation to the aforementioned targets, a critical component is found. A joint that supports high accuracy should provide low clearance between moving surfaces. Then a joint that supports high dynamics should provide low friction. The left side of Figure 7 displays both requirements within the block “joint”, because they are directly related to that component. The arrows show their dependencies to the system requirements and it is possible to follow up to the level of goals. The constraint “contact” illustrates that for a low clearance there is a normal force at the moving surfaces, which generates a friction force. Hence, the smaller the clearance the higher becomes the friction. In conventional joints, this goal conflict is handled by finding the best compromise [30]. The analysis of the model showed that the concerning requirements do not have to be fulfilled at the same time during operation, i.e. when fast moving a low friction is needed, but when assembling the movement is relatively slow and the low clearance is of importance. To allow a time dependent change of joint characteristics new joints with an active adaptability were developed [27, 31].

In later phases sometimes changes appear. For instance, a customer rethinks the cost target: The robot has to be cheaper, otherwise he would back out. Analysing the relations shows that in that case the simplest possibility is to exchange the drives by cheaper (but less powerful) ones. This would mainly effect the “high dynamic” target and to a certain degree the goal “cycle time”. The model allows generating a specific view that makes all these relations transparent and communicable to the customer. Then it would be up to the customer to decide, whether the cheaper robot justifies the loss in cycle time.

One important aspect for working with models especially in collaborative networks is the management of different versions. Modelling software often provides a multi-user repository that allows team members to work concurrently with the same model. Normally a model tree is generated, so that a trunk contains the universally valid versions. The tip version is the actual common state of the project. The different project members generate branches (“private sandbox”) to extend and modify the model according to their subtask. Versions in a branch can be merged at two different manners. “Rebasing” means to integrate the trunk tip version in the branch version and create a new version in the branch. “Reconciling” means to integrate the branch version in the trunk tip version and create a new version in the trunk that constitutes the new common state of the project. Whilst merging two model versions differences can be analysed, i.e. the impact a change in one domain had on model elements also used in another domain are displayed. However, the project management has to ensure a regularly merging of outcomes, an analysis regarding redundant elements, and communication between project members as well as making comments and notes on the made changes. Then it is possible to trace changes and even restore the model to a former condition.

Another opportunity of versioning is the building of variants. Starting from the common state version different branches can be modelled. These branches can detail different possible solutions to figure out the best one. Moreover, it is possible to start from a generic model and detail the different branches according to different development aims, e.g. to develop two different robots for relatively similar boundary conditions but for different tasks.

As pointed out earlier one problem in product development projects is an inadequate documentation of results. Using the models as a developing tool is at the same time a kind of documentation. It gives at least a proper basis for generating the documentation. Modelling software often supports a (semi)automatic document generation. Thence, documents are generated “on the fly” what facilitates this unloved duty of documenting.

5 CONCLUSIONS

Multidisciplinary product development in collaborative networks is a typical working condition for nowadays engineers. After describing the main deficits and dangers of such development situations a modelling approach using the Systems Modelling Language (SysML) is shown. The approach focuses on the generation of a requirements model as a basis for discussion and analysis of the real project aims. Therefore, it discusses the four aspects surrounding, structure, relations, and analysis. A simplified example from the field of parallel robots illustrates the approach and highlights the benefits.

The paper shows that the modelling of requirements is an essential step in the development of complex products. Especially in collaborative networks, it helps to concentrate on and to communicate goals, targets, and requirements. It assists the decision-making processes and makes them more transparent. SysML is a known notation, thence the effort to formalise the model is comparably small, but gains a good exchangeability, e.g. for remote and concurrent working. The generated diagrams are quite easy to understand and hence augment cognition.

6 ACKNOWLEDGMENTS

The authors gratefully thank the German research association (DFG) for supporting the Collaborative Research Centre SFB 562 “Robotic Systems for Handling and Assembly – High Dynamic Parallel Structures with Adaptronic Components”.

7 REFERENCES

[1] Franke, H.-J., Huch, B., Herrmann, C., Löffler, S., (ed.), 2005, Ganzheitliche Innovationsprozesse in modularen Unternehmensnetzwerken, Logos Verlag, Berlin.

[2] Steinmetz, O., 1993, Die Strategie der integrierten Produktentwicklung, Vieweg Verlag, Frankfurt.

[3] Gaul, H.-D., 2001, Verteilte Produktentwicklung – Perspektiven und Modell zur Optimierung, Verlag Dr. Hut, München.

[4] Bender, B., 2001, Zielorientiertes Kooperationsmanagement in der Produktentwicklung, Verlag Dr. Hut, München.

[5] Zanker, W., 1999, Situative Anpassung und Neukombination von Entwicklungsmethoden, Shaker Verlag, Aachen.

[6] Franke, H.-J., Wrege, C., Stechert, C., Pavlovic, N., 2005, Knowledge Based Development Environment.

The 2nd International Colloquium of the Collaborative Research Center 562, Braunschweig, Germany, 10-11 May: 221-236.

[7] Stechert, C., Alexandrescu, I., Franke, H.-J., 2007, Modelling of Inter-Model Relations for a Customer Oriented Development of Complex Products. The 16th International Conference on Engineering Design ICED 07, Paris, France, 28-30 August.

[8] van Beek, T.J., Tomiyama, T., 2008, Requirements for Complex Systems Modelling. The 18th CIRP Design Conference - Design Synthesis, Enschede, The Netherlands, 7-9 April.

[9] Kläger, R., 1993, Modellierung von Produktanforderungen als Basis für Problemlösungsprozesse in intelligenten Konstruktionssystemen, Shaker Verlag, Aachen.

[10] Daenzer, W.F., Huber, F. (Ed.), 2002, Systems Engineering - Methodik und Praxis, Verlag industrielle Organisation, Zürich.

[11] Avgoustinov, N., 2007, Modelling in Mechanical Engineering and Mechatronics - Towards Autonomous Intelligent Software Models, Springer Verlag, London.

[12] Coates, G., Duffy, A.H.B., Whitfield, I., Hills, W., 2004, Engineering management: operational design coordination, Journal of Engineering Design, 15/5: 433-446

[13] Ort A., 1998, Entwicklungsbegleitende Kalkulation mit Teilebibliotheken, Papierflieger, Clausthal-Zellerfeld.

[14] Salustri, F.A., Eng, N.L., Weerasinghe, J.S., 2008, Visualizing Information in the Early Stages of Engineering Design, Computer-Aided Design & Applications, 5: 1-4.

[15] The Official OMG Systems Modelling Language (SysML) site, 2007, http://www.omgsysml.org/

[16] Johar, A., Stetter, R., 2008, A Proposal for the Use of Diagrams of UML for Mechatronics Engineering, The 10th International Design Conference - DESIGN 2008, Dubrovnik, Croatia, 19-22 May: 1287-1294.

[17] La Rocca, G., van Tooren, M.J.L., 2006, A modular reconfigurable software tool to support distributed multidisciplinary design and optimisation of complex products, The 16th CIRP International Design Seminar, Kananaskis, Alberta, Canada,16-19 July.

[18] Jung, C., 2006, Anforderungsklärung in interdisziplinärer Entwicklungsumgebung, Verlag Dr. Hut, München.

[19] Maletz, M., Blouin, J.-G., Schnedl, H., Brisson, D., Zamazal, K., 2007, A Holistic Approach for Integrated Requirements Modelling in the Product Development Process, The 17th CIRP Design Conference - The Future of Product Development, Berlin, Germany, 26-28 March: 197-207.

[20] Franke, H.-J., 1976, Untersuchungen zur Algorithmisierbarkeit des Konstruktionsprozesses, VDI Verlag GmbH, Düsseldorf.

[21] Anggreeni, I., van der Voort, M.C., 2008, Classifying Scenarios in a Product Design Process: a study towards semi-automated scenario generation, The 18th CIRP Design Conference - Design Synthesis, Enschede, The Netherlands, 7-9 April.

[22] Brouwer, M., van der Voort, M.C., 2008, Scenarios as a Communication Tool in the Design Process: Examples from a Design Course, The 18th CIRP Design Conference - Design Synthesis, Enschede, The Netherlands, 7-9 April.

[23] Miedema, J., van der Voort, M.C., Lutters, D., van Houten, F.J.A.M., 2007, Synergy of Technical Specifications, Functional Specifications and Scenarios in Requirements Specifications, The 17th CIRP Design Conference - The Future of Product Development, Berlin, Germany, 26-28 March: 235-245.

[24] Stechert, C., Franke, H.-J., 2007, Requirement-Oriented Configuration of Parallel Robotic Systems. The 17th CIRP Design Conference - The Future of Product Development, Berlin, Germany, 26-28 March: 259-268.

[25] Franke, H.-J., Krusche, T., 1999, Design decisions derived from product requirements, The 9th CIRP Design Conference - Integration of Process Knowledge into Design, Enschede, The Netherlands, 24-26 March: 371-382.

[26] Stechert, C., Franke, H.-J., 2008, Managing Requirements as the Core of Multi-Disciplinary Product Development. The 18th CIRP Design Conference - Design Synthesis, Enschede, The Netherlands, 7-9 April.

[27] Stechert, C., Pavlovic, N., Franke, H.-J., 2007, Parallel Robots with Adaptronic Components - Design Through Different Knowledge Domains, The 12th IFToMM World Congress, Besançon, France, 17-21 June.

[28] Rose, M., Keimer, R., Breitbach, E.J., Campanile, L.F., 2004, Parallel Robots with Adaptronic Components, Journal of Intelligent Material Systems and Structures, 15/9-10: 763-769.

[29] Rose, M.; Keimer, R; Algermissen, S., 2003, Vibration Suppression on High Speed Parallel Robots with Adaptronic Components, The 10th International Congress on Sound and Vibration (ICSV), Stockholm, Sveden, 7-10 Juli.

[30] Otremba, R., 2005, Systematische Entwicklung von Gelenken für Parallelroboter, Logos Verlag, Berlin.

[31] Pavlovic, N, Keimer, R., 2008, Improvement of Overall Performance of Parallel Robots by Adapting Friction of Joints Using Quasi-Statical Clearance Adjustment, Adaptronic Congress 2008. Berlin, Germany, 20-21 May.