LIGHTWEIGHT VISUALIZATION OF SYSML MODELS IN PDM SYSTEMS

Nigischer, Christian; Gerhard, Detlef
TU Wien, Austria

Abstract

Increasing product complexity and diversity especially in case of mechatronic products leads to the need for extended method and tool support during product development. Specialized authoring tools allow the creation of descriptive product models utilizing modelling languages like the Systems Modelling Language (SysML). The high level of complexity also requires a lot of coordination and collaboration effort and the associated data exchange between a variety of stakeholders. Although specifications exist, exchange of SysML model data – in particular of graphical diagram information – is often not possible due to compatibility problems. Hence, a concept for visualizing SysML models in Product Data Management (PDM) systems without using an authoring tool is introduced. A layer model generated by using design phase information delivered by the Requirements, Functional, Logical, Physical (RFLP) product design approach is proposed. Furthermore, various functions of the visualization module are presented. The so-called "Lightweight Visualization" approach shall ease the exchangeability of SysML model data within distributed development networks.

Keywords: Early design phases, Systems Engineering (SE), Visualisation, Systems Modelling Language (SysML), Model data exchange

Contact:
Christian Nigischer
TU Wien
Department of Mechanical and Industrial Engineering
Austria
christian.nigischer@tuwien.ac.at

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 3: Product, Services and Systems Design, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

Innovative products often contain a variety of electronic/electromechanical components like sensors, actors and information processing units. Together with a basic system - in general a mechanical, hydraulic or pneumatic structure, or a combined structure - these components form a mechatronic system (VDI 2206). According to Eigner et al. (2014), the increasing share of mechatronic components including software in products together with the requirement of multimarket adaptability are the main drivers for the increasing product complexity. While the overall product complexity and therefore product development complexity grows, time spans for product development cycles decrease. To handle that increasing complexity with less development time available, special approaches, methodologies and tools are needed. A possibility to deal with the "increasing complexity - less time" dilemma is to create a representative model of the product which allows testing, verification, validation and - where required - simulation activities already in the early product development stages. Model-based systems engineering (MBSE) appears as an appropriate approach to provide such a model covering the whole product lifecycle (Vosgien et al., 2012).
Complex mechatronic systems usually are not developed by a single engineer or engineering department. 20 years ago, Liker et al. (1996) stated that "the importance of communicating information is particularly crucial as design work becomes distributed across multiple players, as it is the case in the automotive industry where large parent companies or Original Equipment Manufacturers (OEMs) are pushing design responsibility onto suppliers." The topic of data exchange between business partners that comes along with distributed engineering design is still an issue today. Ongoing globalization within the value chain leads to more complex networked organizations and processes inside manufacturing enterprises as well as between them within the supply chain. The requirement of interdisciplinary collaboration between all participants across different cultures and time zones becomes more and more important. Furthermore, the results of a study presented by Eigner et al. (2010) show that the integration of heterogeneous data along the product lifecycle still remains a key challenge in the scope of implementing a Product Lifecycle Management (PLM) solution. Especially Product Data Management (PDM) systems which usually serve as product data backbones for PLM activities are affected by data exchange and integration issues.
While exchange of Computer-Aided Design (CAD) data has become easier nowadays due to standardization of data exchange formats like STEP (STandard for the Exchange of Product model data - ISO 10303) or JT (Jupiter Tesselation - ISO 14306) (Katzenbach et al., 2013), exchange and visualization of more abstract information like descriptive SysML (Systems Modelling Language) model data is still more difficult to realize. Although data exchange of SysML content - as SysML itself - is standardized by the Object Management Group (OMG), authoring tool vendors implement standards differently or in various extent. As a result it is often impossible to import models created by authoring tool A into authoring tool B or any other SysML familiar software tool. Another problem is the visualization of SysML model information. Graphical model information is often not exported and therefore not restorable by the target software.
The purpose of the work presented in this paper is to deal with these problems and make SysML model data more accessible outside of authoring tools. After discussing similar research activities in the next section, a concept for a generic visualization of SysML models in PDM systems is introduced. The aim of the concept is to provide an alternative graphical representation of existing SysML models without involvement of authoring systems. Furthermore, potential functionalities which provide information extraction from the visualized model are identified.

2 SYSML SPECIFICATIONS AND RELATED WORK

This section gives an overview of available specifications and related work that addresses SysML data exchange and/or the visualization of SysML content outside of authoring tools.

2.1 OMG SysML specifications

SysML is a general-purpose graphical modelling language, method- and tool-independent, that supports the specification, analysis, design, verification and validation of a wide spectrum of complex systems (OMG SysML Specification 1.4, 2015). The current SysML 1.4 specification is available since

September 2015 in its final version and offers some changes compared to version 1.3. Weilkiens (2014) gives a summary of the most important changes like the addition of an «ElementGroup» Stereotype, the revision of the viewpoint/view concept or the introduction of a new model element called "boundReference". The most relevant change for this work is the replacement of the Diagram Interchange (DI) specification by the Diagram Defintion (DD) 1.1. The Diagram Defintion specification (OMG DD 1.1, 2015) provides the capabilities that allow the exchange of graphical notations like SysML diagrams. Therefore, the DD uses the XML Metamodel Interchange (XMI) format as basis for diagram data exchange. XMI is also used for non-graphical model data exchange. Thus, both kinds of information can be merged in one XML structured export file. As Weilkiens (2014) mentioned, there was no authoring tool available in 2014 which had implemented the previous DI specification. Hence, the exported XMI files were restricted to model data, not containing any diagram specific information.

2.2 SysML model data export and visualization

Friedenthal et al. (2015) specify 3 ways of data exchange between tools in a systems development environment:

- Manual exchange involving re-entering the data from one tool into another tool;
- File-based exchange using a proprietary file format or standard exchange format (e.g., XMI); or
- Interaction-based exchange using Application Programming Interfaces (APIs)

Obviously, manual data exchange is irrelevant for this work. In fact SysML data shall be extracted from a model file and properly visualized without re-entering any data. The approach using APIs for direct communication between different software tools requires adaptions for every participating software tool. This work focuses more on the exchange via standard formats with very low adaption effort needed to offer as much freedom and flexibility as possible regarding authoring tool selection.
Eigner et al. (2012) present a data model that allows the integration of Functional Product Descriptions into PDM systems. In addition to the existing requirement and product structures (also called BOM - Bill of Materials), functional and logical elements are established in the PDM data scheme to support the RFLP (Requirement, Function, Logical, Physical) product design approach. The first phase of the RFLP approach translates customer needs into requirements and technical specifications, which are the initial point for the product development activities. The second phase defines product functions uncoupled of concrete technical implementations. The third phase determines logical solution elements that implement the functions defined in the previous phase. The fourth and last phase of the RFLP approach represents the domain-specific development of the physical product elements. An assignment of the RFLP design phases to the well-known V model can be found in the work of Eigner et al. (2012). Kleiner and Kramer (2013) offer a more detailed description of the RFLP approach and its application.

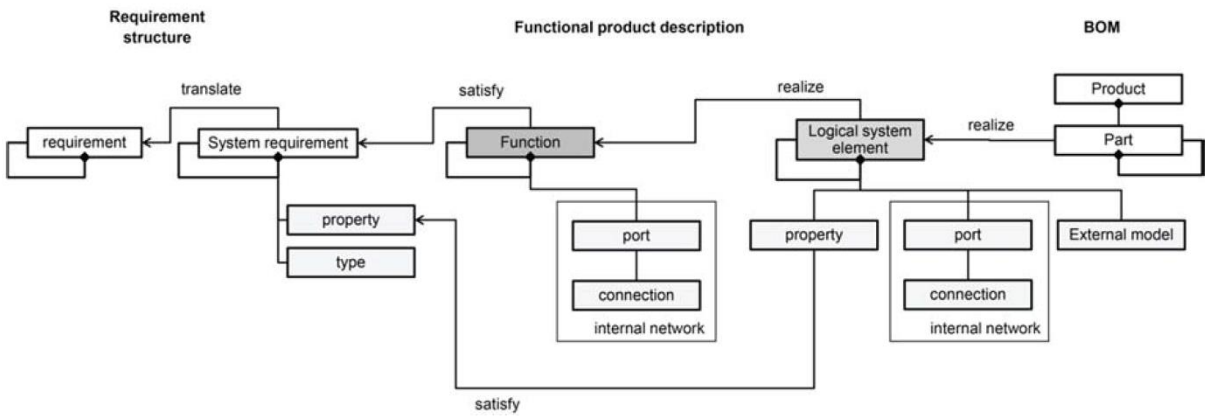

Figure 1. Simplified data schema of the Functional Product Description model (Eigner et al., 2012)

The new elements in the data model of Eigner et al. (2012) shall bridge the gap between requirements and physical product structure. Elements of the already existing data structures are connected via 3 different types of connections with the new elements: Hierarchies, typed internal connections via ports and cross-references with allocations between different model elements. The recommended data

structure depicted in Figure 1 is very similar to a SysML model structure. Therefore, it is an eligible candidate for storing SysML data exported from an authoring tool in a PDM system.
Trase and Fink (2014) introduce an Interactive Visualization Engine for SysML Tools (InVEST) that is able to extract SysML model data from reports created by an authoring tool using the Velocity Template Language (VTL), which is a description language to define text file templates. The extracted data is abstracted to generate meaningful views of the model data even to stakeholders that are not familiar with the SysML notation. "While free model viewing tools exist, InVEST abstracts the relevant subsets of model data into a variety of interactive visualizations that do not require prior knowledge of SysML" (Trase and Fink, 2014). Figure 2 shows the possible visualization options like bubble charts or list hierarchies. Abstraction is also part of the concept presented in this paper. Full SysML notation is not always necessary respectively possible in alternative forms of data representation.

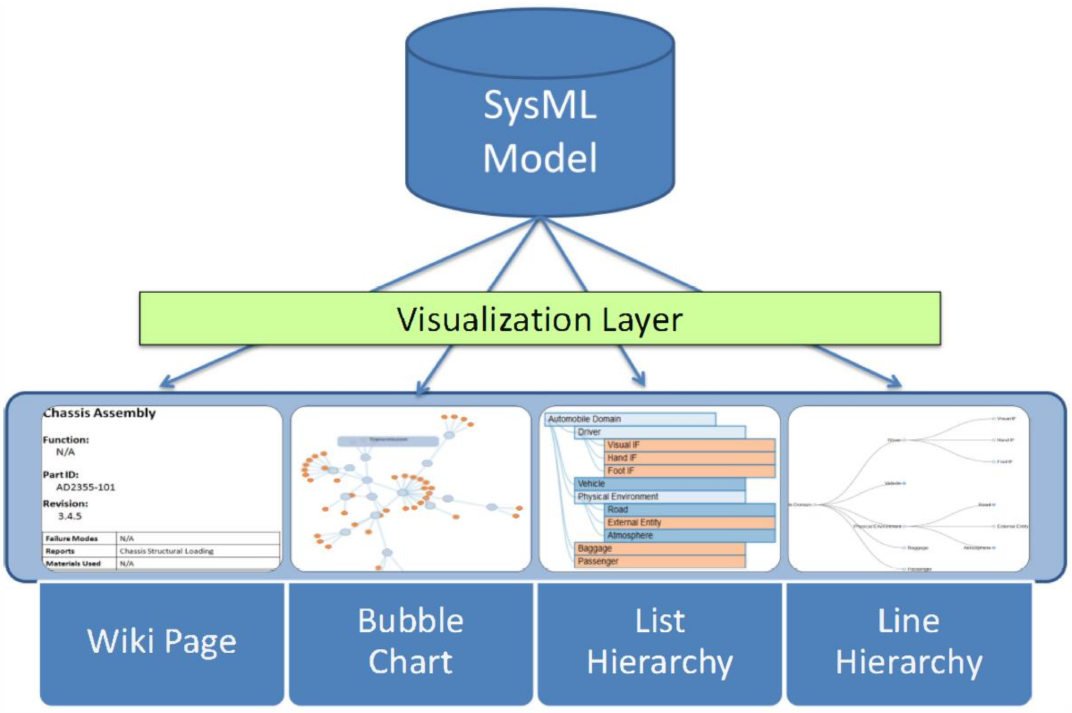

Figure 2. InVEST visualization overview (Trase and Fink, 2014)

Similar to Trase und Fink, Sindiy et al. (2013) state that the effective communication of output diagrams is one of the barriers for the success of MBSE. The information carried by SysML models has to be presented properly to all involved stakeholders. While Trase und Fink use simplification and abstraction to make SysML content more accessible, Sindiy et al. introduce high-level guidelines for visual presentation of MBSE efforts. These guidelines base upon techniques from the information visualization (InfoVis) discipline and shall help to improve the readability of model diagrams. Examples for such InfoVis considerations are improved layout and alignment of diagrams, the use of colours, adjusted font types and sizes, adapted proportionalities, revised data selection for different viewpoints, and so on. Concerning automated diagram generation, Sindiy et al. mention that many of today's MBSE tools already contain toolbars and/or user defined "style sheets" to implement some of the mentioned InfoVis functionalities, but "the existing automation capabilities fall short of being able to automatically generate complete diagrams and as such, some manual user interaction is still required and needs to be planned for." In general, the graphical representation of a comprehensive SysML model as aspired in this work will be confusing for the observer because of too many model artefacts. Therefore, functionalities that limit the amount of information provided in a specific view are needed. Sindiy et al. suggest to use auto-layout features with additional rules for dynamically creating output views.
The work of García (2013) addresses - among other aspects - the measurement and visualization of SysML model data created by the authoring tool Enterprise Architect from Sparx Systems. Therefore, a software tool called Visualization of SysML Meta-model Architecture (VoSMA) was designed and implemented. VoSMA is able to extract the SysML data from XMI files stored by Enterprise Architect using a Java based XML parser in combination with a SysML element reader module. As a next step,

the meta-model of the investigated project is reconstructed by processing the extracted SysML raw data. Finally, the abstracted model data is visualized by using different occurrences of tree models. Visualization can also be accomplished by utilizing a third-party tool that processes output data from the VoSMA tool. Figure 3 gives an overview of the approach proposed by García.

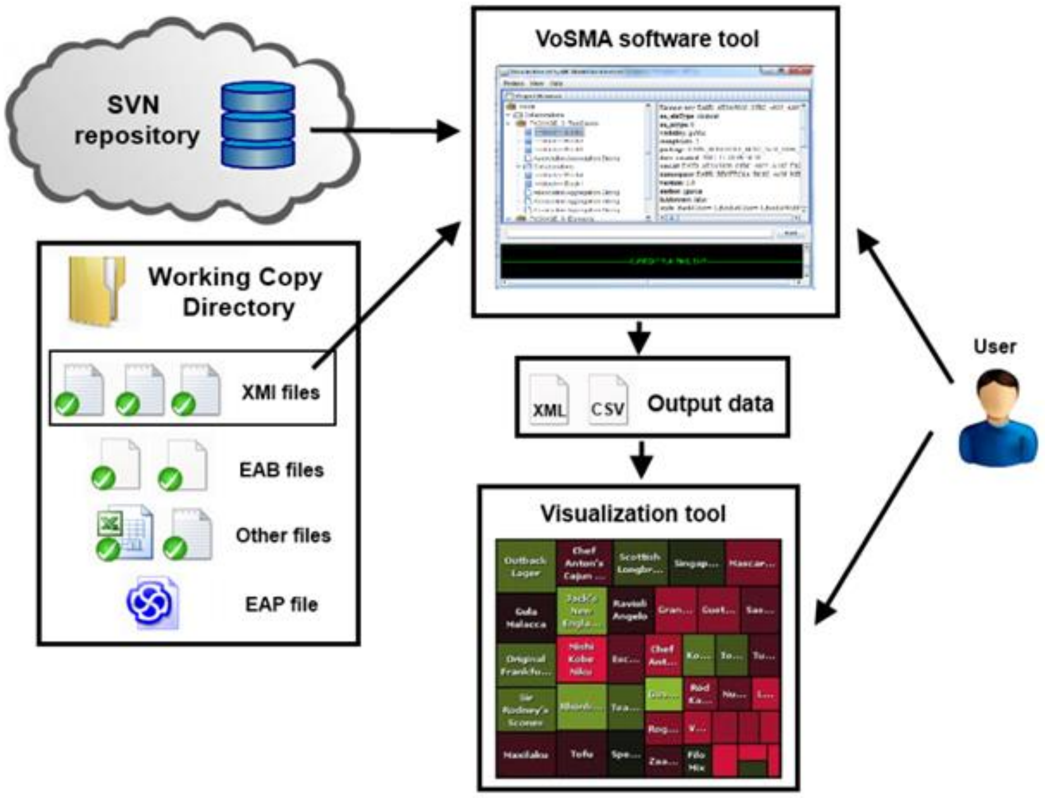

Figure 3. Overview of the proposed approach using the VoSMA tool (García, 2013)

3 SYSML MODEL VISUALIZATION IN PDM SYSTEMS

As mentioned in Section 2, several authors highlight the importance of appropriate graphical representation of model data. The modelling itself is usually supported by specific authoring tools, for instance CAD tools or modelling suites for systems/software designing. The created models are typically stored as proprietary model files in a PDM system that functions as a product data backbone. Among other functionalities, the PDM system ensures data access and version control. In case of CAD data that is stored in an up-to-date PDM system, integrated viewer modules offer a graphical representation of the part in the PDM system. For instance, the PDM system Teamcenter (Siemens PLM, 2016) is able to display a part in a 3-dimensional view, if a corresponding Jupiter Tessellation (JT) file of the part is available. Eigner et al. (2010) describe the JT data format as a lightweight data format that functions as a primary data carrier within virtual product development alternatively to native CAD files. Meanwhile the JT format is specified in the ISO 14306 (2012) standard. As already mentioned in Section 2.1, SysML interface specifications published by the OMG do exist, but are currently not fully implemented by the SysML authoring tool vendors. Hence, the exchange of model and diagram data between different software tools is often impossible. From an overall perspective there are different ways to attain a graphical representation of SysML models in a PDM system:

- Full integration of SysML authoring tool and PDM system;
- Specialized plug-ins or modules enable a PDM system to display/allow manipulations of models created by a specific authoring tool;
- Use of a more general plug-in or module that is not linked to a specific authoring tool for the approach of "lightweight" visualization of a model

A full integration of SysML authoring capabilities in a PDM system means that the PDM system literally becomes an authoring system itself. All functions for creating and manipulating SysML content have to be implemented in the PDM system. Furthermore, not only model files as a whole have to be managed, but every single model artefact. All model data - even on artefact level - is directly accessible in the PDM system. Another advantage is that PLM functionalities, which are already implemented in the PDM system, e.g., version control and the use of workflows to control processes, can be applied on artefact level. With all authoring tool functionalities implemented the visualization of models and their related diagrams in the PDM system is not a problem at all. The challenge of exchanging SysML content with different tools using divergent export formats persists. The high degree of dependence of the integrated authoring PDM tool can be a problem regarding system reliability. If a system failure occurs, both parts of the system are equally affected. Another disadvantage is that benefits of the integrated solution only can be utilized if the PDM system and the implemented modelling tool both meet predefined requirements. If the implemented modeller does not fulfil all needs and the additional use of an external authoring tool is needed, the potential of the integrated solution is lost.
Specialized plug-ins or modules provide at least a subset of the functionalities of an authoring tool directly embedded in the environment of the PDM system. The difference to full integration is that not every single model element is stored and managed in the PDM system, but the proprietary file format of the corresponding authoring tool is used to store the model information within the PDM system. This file-based approach implicates that not all data on artefact level is available for other PDM functionalities but only the relevant metadata. Because this kind of plug-ins or modules work with the proprietary file formats, visualization of SysML model diagrams created by the corresponding authoring tool - and only the corresponding tool - is possible. Compared to full integration, this approach using specialized plug-ins is more flexible regarding authoring tool selection, if appropriate plug-ins are already available. Otherwise, the development of a new plug-in may cause noticeable costs. Another cost factor may be licensing costs, if different authoring tools and proper PDM plug-ins are needed.
The third option is a general plug-in or module that is not designed for a particular authoring system or file format. In contrast to specialized plug-ins a universal plug-in cannot handle proprietary file formats but is able to process authoring tool export files using a standardized exchange format, for instance the XMI format. The main problems of this approach, deviating implementations of the existing OMG standards and the lack of integration of graphical representation information in such export files, were already addressed in Section 2. The essential advantage of this approach is that SysML models created in different authoring tools can be visualized using only one plug-in. This can be helpful especially for distributed development conglomerates with a highly diverse development tool landscape. The concept for the realization of such a general plug-in for PDM systems is introduced in the following sections.

4 LIGHTWEIGHT VISUALIZATION OF SYSML MODELS

The developed concept described in the following sub-sections consists of two essential parts which are explained in the three following steps:

- The basic approach that gives a rough overview of the involved system components,
- the XMI interpreter that extracts model information from XMI files (1st major part), and
- the visualization module that creates graphical representations of SysML model data (2nd major part).

4.1 Basic concept

Figure 4 depicts the basic approach using a XMI interpreter in combination with a visualization module to build generic graphical representations of SysML XMI formatted models in a PDM system. The concept works similar to the well-established practice utilizing JT files generated by CAD authoring tools during model saving transactions additionally to the native CAD files to allow graphical model representation in the PDM environment. A number of different SysML authoring tools is used to create SysML models. Models are stored in the PDM data storage system, symbolized by the "Data storage" box, using the proprietary file format of the particular tools. In addition, each authoring tool exports the model as XMI file with vendor specific structure. The XMI model file is also stored in the PDM system linked with the corresponding proprietary model file. Stored XMI files can be sent to a XMI interpreter that extracts the available model data and transforms it to a format that is more suitable for visualization matters than serialized XMI model data. The converted model data is then forwarded to the visualization module implemented in the PDM system, which generates a generic graphical representation of the model and ideally provides further useful functionalities like search functions or impact analyses. XMI interpreter and visualization module do not have to be separate programs, they can also be combined in an integrated module. The term "lightweight" visualization was chosen by analogy with the JT data format, which is often designated as lightweight data format.

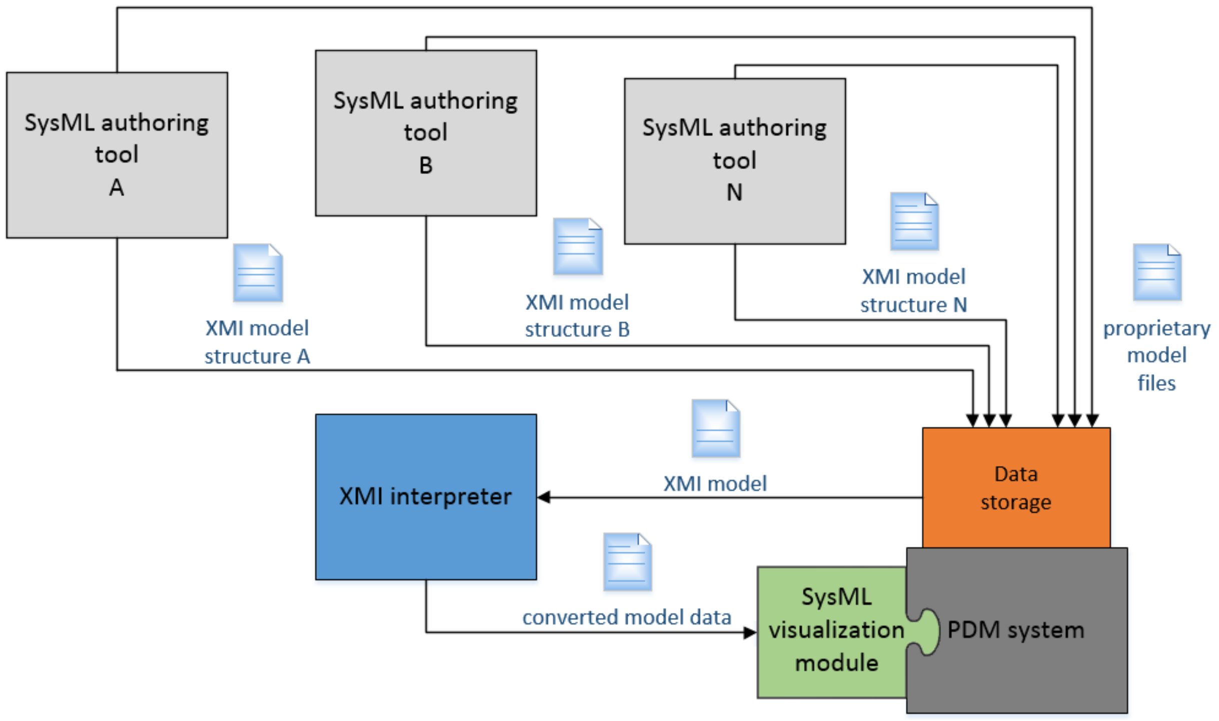

Figure 4. Basic approach of using a lightweight visualization in a PDM system

4.2 XMI interpreter

First, the XMI interpreter functions as a parser. The serialized model information is extracted from the XMI input file. Because of the already mentioned different implementations of OMG XMI specifications by various vendors, the XMI interpreter has to be able to handle different variations of XMI files, like namespace or structural deviations. The extracted model information has to be transformed into data structures that are easier to handle than the fragmented data representation in a XML structured XMI file. In the first prototypes a self-programmed parser was used to extract the SysML model artefacts. The model artefact information was temporarily stored as 2 different types of objects - element and relationship objects, supplemented by their attributes - and forwarded to the visualization module. It has to be mentioned that proof of concept tries were just made with rather simple SysML models, so it may will be necessary to introduce some additional object types. For further work, already established XML parsers can be considered to replace the self-programmed parser to save some time and effort. A selection of Java XML parser APIs is introduced in the work of García (2013).

4.3 SysML visualization module

The visualization module processes the data containing objects created by the XMI parser with the aim to generate a graphical representation of the SysML model and display it in the environment of the PDM system. Some authoring tools, like MagicDraw (No Magic, 2016), are able to include graphical diagram information in the export XMI files. If that information is provided, SysML diagrams with the corresponding content can be reconstructed and depicted in the familiar 2D diagram style. Do XMI model files not include such graphical information, an alternative generic graphical representation can be utilized to visualize the SysML models. Ideally, the visualization module is able to do both but this work focuses on the implementation of a generic model representation, which is described in more detail in the following sub-section.

4.3.1 *Generic SysML model representation*

The aim of the generic approach is to provide a visualization of SysML models stored in XMI files, even without diagram information available. The main question is how models can be illustrated without any layout and positioning information.

First of all, a minimum of structural information is needed to form any kind of graphical representation. In this concept, the required information is provided by the use of the RFLP approach during the design process of the SysML model. More precisely, newly added SysML elements have to be tagged to associate them to one of the RFLP design phases if such an identification mark does not already exist. For instance, requirements can be easily identified because of the «Requirement» stereotype. Other SysML modelling elements like blocks or actions have no implication for belonging to a specific phase. Hence, a new indicator has to be applied to assign that more general elements. As several authors - e.g. Weilkiens (2008) or Chiron and Kouiss (2007) - describe, the SysML profile, which contains the standard set of SysML stereotypes, can be extended by adding new stereotypes. In case of this concept, the addition of «Function», «Logic» and «Physical» stereotypes would allow the needed assignment. Although adding new stereotypes is considered as the simplest way, other tagging possibilities are also possible - for instance the definition of a distinguishing attribute for the affected SysML artefacts.

With the information which model elements are linked to which RFLP design phase and the relationships between the elements within the model, a layer model can be established. An early mock-up is depicted in Figure 5. Each RFLP design phase can be pictured by a set of layers. The upper set of layers (R) is a representation of the Requirements design phase, while the middle set of layers (F) is associated with the Function design phase and the bottom set of layers (L) is used for the Logic phase. There are no layers for the Physical design phase depicted because the physical elements are often not present in the SysML model but stored as product structures in the PDM system. If artefacts assigned to the physical design phase are present in the XMI model file, or if the corresponding product structure is extracted from the PDM system, these P-layers can also be added to the visualization model. Every layer represents a hierarchical level of the model elements of a design phase. Every element that has no hierarchical connected parent element is placed at the top-layer of the corresponding design phase. Elements that are linked to parent elements with hierarchical connections, like compositions or containment links, are placed as child elements a layer below their parent elements. The layers below the top-layer are declared as sub-layers. An arbitrary amount of sub-layers is possible for every RFLP design phase. While the hierarchical connections structure the elements within a design phase, cross-references with allocations are able to link model elements within a design phase as well as elements assigned to different design phases. Following the work of Eigner et al. (2012), which was shortly addressed in Section 2.2, the third possible connection type are internal connections between elements via ports. These connections are not depicted in Figure 5, but can be implemented as well. The main problem that has to be solved is the positioning of the model elements on the layers in a smart way, so that the visualized model can be interpreted by the observer.

4.3.2 *Visualization module functionalities*

To make the comprehensive visualization of an extensive model with a lot of elements and relationships between them manageable and to utilize potential benefits, the visualization module has to provide a spectrum of functionalities.

SysML models can contain a large amount of information which has to be properly presented to the viewer. The usual approach during modelling using an authoring tool is to create multiple diagrams or views which each shows a fraction of the model information only. For a comprehensive model visualization smart data representation is even more important to prevent information overload on the screen. Therefore, intelligent automated functions - for instance zoom level dependent data display - supplemented by user handled options like filters to display only selected element/relationship types and highlighting functions have to be implemented. Another essential capability of the visualization module are search functions to allow fast localization of model artefacts. More advanced functionalities like traceability and impact analyses can bring additional value. For instance, it should be possible to highlight all connected requirements of any selected model element, including the paths of model connections that link the selected element and the corresponding requirements. Furthermore, functions to create 2D projections of selected areas/hierarchies of the 3-D layer model can reduce the spatial complexity, where needed. In another stage of extension, the visualization module ideally is also able to redraw the diagrams generated in an authoring tool, if the graphical diagram information is enclosed in the XMI model file. Cross references between the diagrams and the generic layer model can make it handy to use both visualizations simultaneously.

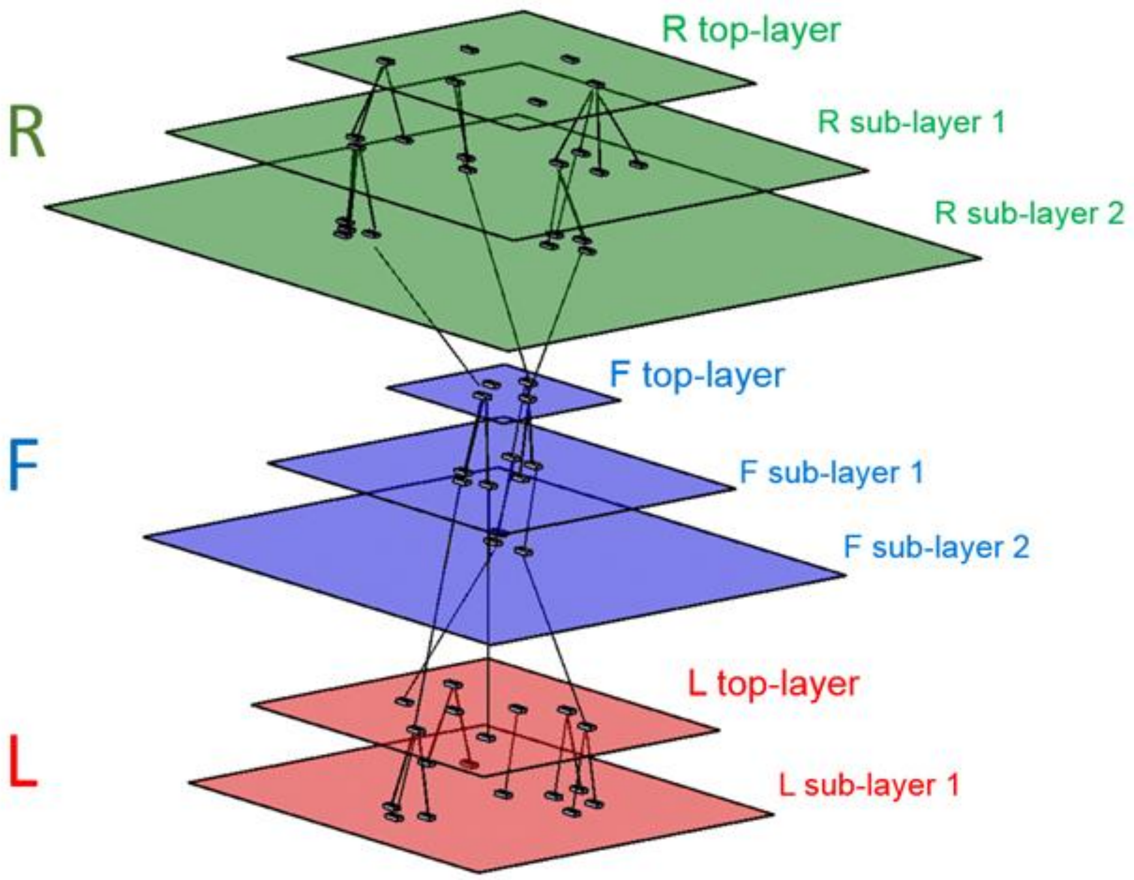

Figure 5. Mock-up of generic 3-D visualization of a SysML model

5 CONCLUSION AND OUTLOOK

The introduced concept provides an approach for visualizing SysML models created by different authoring tools using a 3-D layer model which uses the RFLP product design approach as basis. The concept in its current state does not consider all details, especially the positioning of the elements in a smart way so that the expressiveness and interpretability are as high as possible, is still a challenge. Another problem is to make the XMI interpreter capable to handle as many output model formats from different SysML authoring tools as possible.

Of course, the generic visualization will not be able to offer all intricacies of a manual created SysML diagram, but it can be helpful to give an overview of the overall system. Moreover, it offers a visualization for exported XMI models which do not contain any diagram information. In the end, the implemented functionalities for data search, filtering and options for on-screen representation of the model will determine the usefulness of the lightweight vision module.

The next steps will be to assess existing XML parsers and related data visualization approaches to find realizable solutions for the 2 main challenges - generic element positioning and multi-authoring tool compatibility of the XMI interpreter. After detailing the concept, a prototypic implementation shall prove the feasibility of this generic visualization efforts using several SysML models as examples in a case study.

If the mentioned problems can be solved and the described concept turns out as applicable, the lightweight visualization approach can be beneficial for companies that participate in complex development environments with lots of stakeholders and different IT landscapes involved. The exchanged SysML model data can be visualized tool-independent and without the need of graphical diagram information.

REFERENCES

Chiron, F., Kouiss, K. (2007), “Design of IEC 61131-3 Function Blocks using SysML”, *Mediterranean Conference on Control & Automation*, Athens, Greece, June 27-29 2007, IEEE, DOI 10.1109/MED.2007.4433695

Eigner, M., Gerhardt, F.J., Gilz, T., Handschuh, S. (2010), "Neutral data formats in product development: From use cases to a requirements portfolio", *11th International Design Conference*, Dubrovnik, Croatia, May 17-20 2010, The Design Society, Glasgow, pp. 1471-1480.

Eigner, M., Gilz, T., Zafirov, R. (2012), "Proposal for functional product description as part of a PLM solution in interdisciplinary product development", *12th International Design Conference*, Dubrovnik, Croatia, May 21-24 2012, The Design Society, Glasgow, pp. 1667-1676.

Eigner, M., Roubanov, D., Zafirov, R. (2014), *Modellbasierte virtuelle Produktentwicklung*, Springer Vieweg, Berlin. DOI 10.1007/978-3-662-43816-9.

Friedenthal, S., Moore, A., Steiner, R. (2015), *A practical guide to SysML: The Systems Modeling Language*, Morgan Kaufmann, Boston. ISBN 978-0-12-800202-5.

García, J. (2013), *Visualization of SysML Project Meta-Model Architecture and Evolution*, Master's thesis, Delft University of Technology, Software Engineering Research Group, Department of Software Technology

ISO 14306 (2012), "Industrial automation systems and integration - JT file format specification for 3D visualization", Beuth, Düsseldorf

Katzenbach, A., Handschuh, S., Vettermann, S. (2013), "JT Format (ISO 14306) and AP 242 (ISO 10303): The Step to the Next Generation Collaborative Product Creation", *IFIP TC 5 International Conference, NEW PROLAMAT*, Dresden, October 10-11 2013, Springer, Berlin, pp. 41-52. DOI 10.1007/978-3-642-41329-2_6

Kleiner, S., Kramer, C. (2013), "Model Based Design with Systems Engineering Based on RFLP Using V6", *23rd CIRP Design Conference*, Bochum, Germany, March 11-13 2013, Springer, Berlin, pp. 93-102. DOI 10.1007/978-3-642-30817-8_10

Liker, J.K., Sobek II, D.K., Ward, A.C., Cristiano, J.J. (1996), "Involving Suppliers in Product Development in the United States and Japan: Evidence for Set-Based Concurrent Engineering", *IEEE Transactions on Engineering Management*, Vol. 43 No.2, pp. 165-178. DOI 10.1109/17.509982

No Magic (2016), "MagicDraw", [online] No Magic, Inc. Available at: https://www.nomagic.com/products/magicdraw.html (accessed November 12 2016).

OMG DD 1.1 (2015), "OMG Diagram Definition (DD) Version 1.1", [online] Object Management Group (OMG), Available at: http://www.omg.org/spec/DD/1.1 (Accessed November 30 2016).

OMG SysML Specification 1.4 (2015), "OMG Systems Modeling Language Version 1.4", [online] Object Management Group (OMG), Available at: http://www.omg.org/spec/SysML/1.4 (Accessed December 05 2016).

Siemens PLM (2016), "Teamcenter", Available at: [online] Siemens Product Lifecycle Management Software Inc., Available at: www.plm.automation.siemens.com/en_us/products/teamcenter/ (Accessed November 04 2016).

Sindiy, O., Litomisky, K., Davidoff, S., Dekens, F. (2013), "Introduction to Information Visualization (InfoVis) techniques for Model-Based Systems Engineering", *11th Annual Conference on Systems Engineering Research*, Atlanta, Georgia, March 19-22 2013, Elsevier. DOI 10.1016/j.procs.2013.01.006

Trase, K., Fink, E. (2014), "A Model-Driven Visualization Tool for Use with Model-Based Systems Engineering Projects", *IEEE Aerospace Conference*, Big Sky, Montana, March 1-8 2014, IEEE. DOI 10.1109/AERO.2014.6836268

VDI 2206 (2004), "Entwicklungsmethodik für mechatronische Systeme - Design methodology for mechatronic systems", Beuth, Düsseldorf

Vosgien, T., Nguyen Van, T., Jankovic, M., Benoît, E., Bocquet J.-C. (2012), "Towards Model-Based System Engineering for Simulation-Based Design in Product Data Management Systems", *IFIP International Conference on Product Lifecycle Management*, Montreal, July 9-11 2012, Springer, Berlin, pp. 612-622. DOI 10.1007/978-3-642-35758-9_55

Weilkiens, T. (2008), *Systems Engineering with SysML/UML: Modeling, Analysis, Design*, Morgan Kaufmann, Burlington. ISBN 978-0-12-374274-2.

Weilkiens, T. (2014), *What's new in SysML 1.4 - Overview*. [online] MSBE4U - Tim Weilkiens. Available at: http://model-based-systems-engineering.com/2014/09/16/whats-new-in-sysml-1-4-overview/ (Accessed November 24 2016).