# Introducing Systems Engineering Views in Product Lifecycle Management

Frédéric Autran
CASSIDIAN
1 Boulevard Jean Moulin
78996 Elancourt - France
frederic.autran@cassidian.com

Dieter Scheithauer
CASSIDIAN
Rechliner Str.
85077 Manching - Germany
dieter.scheithauer@cassidian.com



**Abstract.** Product Lifecycle Management demands to integrate all engineering data of a product or service in order to provide full traceability of dependencies between information of different types and from various sources. Current Product Data Management solutions set the focus mainly on data representing physical items. Version Management applications originating from software engineering environments are frequently used also for storing systems engineering data. However, this data management landscape is not adequate for the central role of systems engineering driving both, the development of physical system elements and of system elements implemented by software.
This paper describes how all product data may be integrated with adequately emphasizing the role of systems engineering. It is based on principles EADS is establishing for an EADS-wide standardized but versatile approach appropriate for the wide range of products and services offered by the EADS Divisions Airbus, Astrium, Cassidian and Eurocopter. Currently the defined and agreed concepts are further progressed and implemented.

## Introduction

In times with less sophisticated systems engineering methods, Product Lifecycle Management (PLM) was mainly accomplished by configuring documents and generating configuration baselines. The ISO standard on configuration management ISO 10007 (ISO 2003) still allows this as a valid approach for configuration management.

In two domains handling information by just documents proved to be insufficient, and a more detailed level of configuration control materialized. The drawing set for a product consists of a number of individual drawings that are each configured separately at first. Similar solutions arose for managing source code files that together represent a complete software product.

From both origins the advance in information technology has led to powerful tools for supporting Product Data Management (PDM) on one hand, or Software Version Management (SVM) on the other. For more details on the evolution of tools see the book *Implementing and Integrating Product Data Management and Software Configuration Management* (Crnkovic, Asklund and Persson Dahlqvist 2003). PDM tools cover structural, mechanical and electrically engineering. They support assembly and in-service maintenance. SVM tools concentrate on software, but are also used in a systems engineering context.

The configuration management capabilities of PDM and SVM tools evolved into different directions. For example, PDM tools are targeted to master product variants. In software engineering, concurrency of development activities demands branching and merging capabilities applied to textual data. Consequently, SVM tools excel in this.

Of course, as more information is expressed in structured data formats, today further specific data management tools exist like for requirements management. Due to the narrow focus of these tools they are disregarded here as they miss the configuration management capabilities to support PLM as a whole.

Another trend is the standardization of systems engineering data representation for example ISO 10303-AP233 (ISO 2010) or ReqIF (OMG 2011). But such common data interchange formats are still not applied on a large scale. Therefore no assumption was made regarding the systems engineering data representation to be used.

**PLM Integrating Development and Assembly:** The fragmentation in PDM and SVM tools is a burden for systems engineering to provide an integrated overall PLM. For explanation, Figure 1 shows schematically the complete product evolution covering engineering and assembly activities.

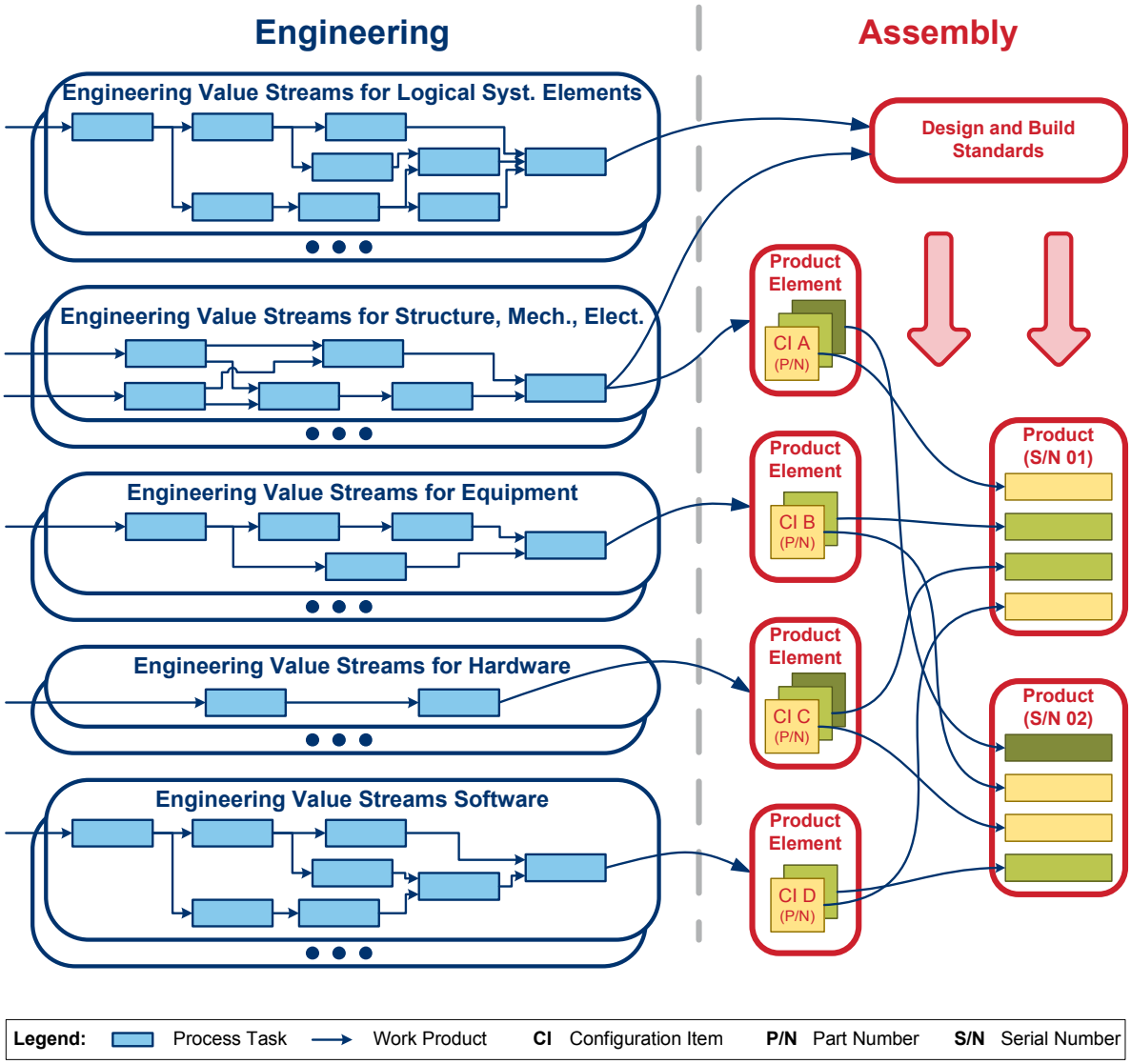


Figure 1. Product Evolution

In engineering, coupled, but widely independently managed value streams exist for the development of

- logical system elements on the upper levels of the system architecture
- system elements dealing with structure, mechanics and electrics down to the implementation level
- installed hardware like sensors, computers and actuators
- software elements implementing a major portion of the overall product functionality

In assembly, individual products are built taking all the product elements needed from the storage. The solution space defining valid product configurations is defined by design and build standards. The individual product elements are identified by part numbers. Generally, they are outcomes of the engineering value streams on the implementation level. In contrast, the design and build standards are merely a result of the systems engineering value streams dealing with the logical system elements, although this information may be provided by drawing sets according to traditional practices.

From this overall view, classical PDM is mainly concerned with supporting assembly. It controls the engineering value streams for structural, mechanical and electrical development. To some extent also engineering value streams for equipment and hardware development may be controlled by PDM as well. Software elements are considered to be delivered as completed configuration items, but their engineering is managed somewhere else. In a similar way, the evolution of the design and build standards is only partly controlled by PDM. It is assumed that the systems engineering value streams work properly under an external management scheme.

These practices limit the overall PLM capabilities of classical PDM since the importance of systems engineering for the generation of design and build standards is disregarded. For enabling control of the complete product evolution in a unique context, systems and software engineering environments need to be integrated into classical PDM applications for an efficient PLM.

**The Vision:** Enable integrated Product Lifecycle Management of all product data by proper integration of systems engineering with Product Data Management solutions.

To accomplish this vision, a particular working group has been established in the context of an EADS-wide effort for PLM harmonization (Mondon 2009). This working group is staffed with systems engineering specialists from all EADS divisions in order to define a systems engineering interface policy for EADS. The solution combines a standardized approach providing the flexibility to adequately support the wide range of products and services developed, manufactured and maintained by EADS.

This paper explains the principles and the proposed approach to better integrate systems engineering into PLM. On the basis of an abstracted view on data management, an integration concept is defined. The fundamental decision is to allow multiple systems engineering environments to be integrated with a single Master Product Definition application. The objectives applicable to the Master Product Definition are derived. It is explained how some pitfalls associated with the chosen approach can be avoided. Finally, the standardized data model for representing systems engineering information in the Master Product Definition is described.

## Application Architecture for Data Management

Traditionally, systems engineering tools are bundling a user interface with a proprietary data model. The integration of multiple systems engineering tools becomes a daunting task as the number of possible tool interfaces increases following a quadratic function with the number of tools. E.g., for integrating the n-th tool, n-1 additional interfaces need to be considered. To segregate the application data processing from the data management as shown in Figure 2 leads to a more scalable solution. Each tool just communicates with the common data repository by dedicated import/export interfaces. Data exchange formats like ISO 10303-AP233 (ISO 2010) and ReqIF (OMG 2011) are principally supportive to define the interfaces.

The data management is further split up in two layers. One is concerned with the evolution of individual work products, e.g. the essential artifacts generated by the systems engineering process. The other deals with the evolution of system releases capturing all system elements up to the overall product or service. The reasons for this differentiation will become clear with the following description of each layer and their association with the systems engineering environment domain and the Master Product Definition domain.

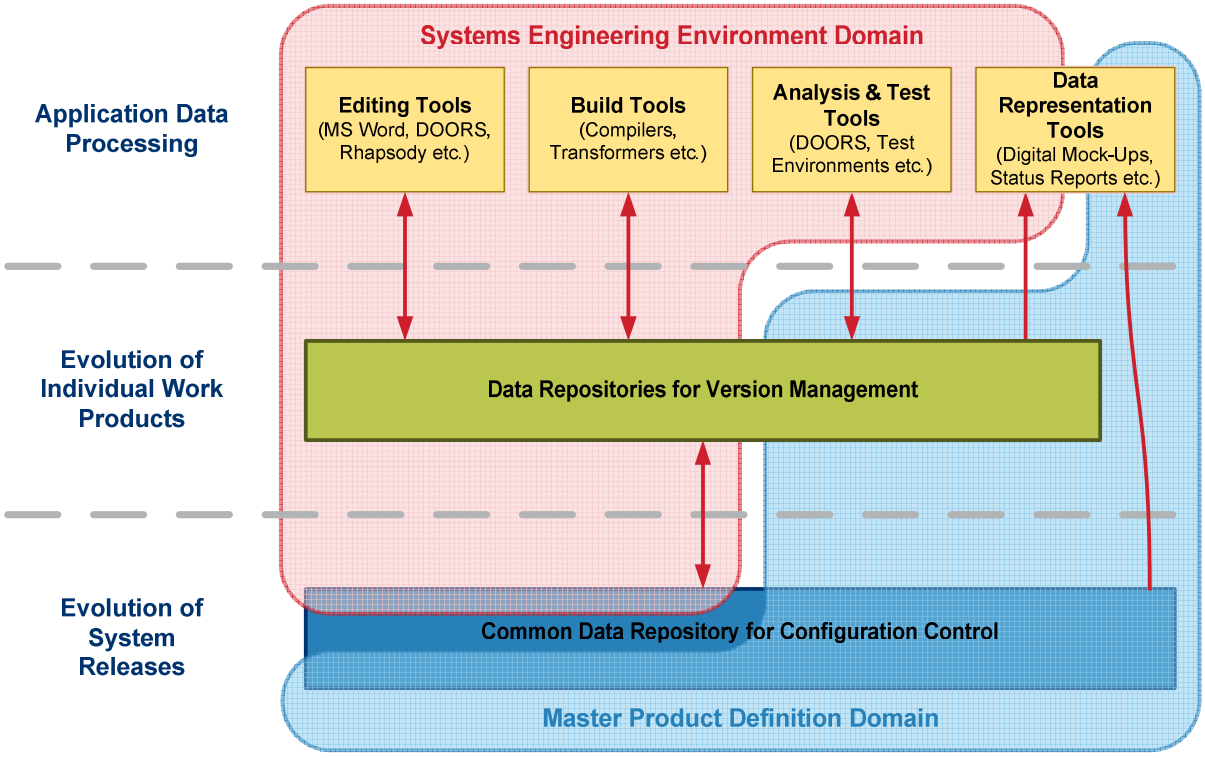


Figure 2. Application Architecture for Data Management

**Application Data Processing.** The application data processing layer comprises the user interfaces and all the underlying processing for user interaction. Editing tools are used to convert human thoughts into digital data. Build tools operate on digital data generating further digital data like compilers or other transformations do. Analysis and test tools are applied to evaluate the quality. For this purpose, they process already existing digital data, and they generate further digital data containing the evaluation results. Of course, commercial systems engineering tools may satisfy functionality of more than just one category. All three kinds of mentioned tools alter data in the data management repositories.

Compared to that, data representation tools access the data management repositories in read-only mode. Data representation tools may extract data from several data repositories to seamlessly integrate them for browsing, navigation, and configuration status accounting. They are well suited to expose systems engineering result to the world outside the systems engineering team as they do not need to include all the intricacies of a user interface dedicated to the editing and manipulation of systems engineering data.

**Evolution of Individual Work Products.** In this layer, the content of all work products, and optionally further supporting data are stored. When data formats are directly readable and interpretable by humans, long-term storage and archiving can be ensured without bothering about obsolescence of any application data processing tool.

For managing concurrency in systems engineering, branching and merging capabilities are important. Maximum control is achieved when two different changes can be generated in isolation followed by later merging both changes consecutively into the main trunk. Usually, SVM tools provide appropriate functionality to accomplish this.

**Evolution of System Releases.** This data management layer hosts all the relations between all the work products. This means that the work product content of each configuration baseline is defined. As far as systems engineering is concerned, aggregating the information up for all system elements over the whole system architecture leads ultimately to the design and build standards.

At least per product or service, the evolution of system releases should be managed in one repository only in order to maximize the efficiency of PLM. Of course, it makes also sense to strive for an enterprise-wide solution where the evolution of the system releases of all products and services offered by the enterprise are managed. Standardization and re-use of system elements over the whole enterprise including product line management may then be managed within a single application.

**Systems Engineering Environment Domain.** In the following, we refer to the term systems engineering environment in order to denominate any tools or tool suites used for performing systems engineering. In its modest form, a systems engineering environment may consist of one or more non-integrated application data processing tools. Information may be stored in tool proprietary data formats.

At the other end of the spectrum, a fully integrated systems engineering tool suite with a common data repository controlling the evolution of work products as well as the evolution of product releases. However, as not all engineering is systems engineering, it is not expected that a systems engineering environment will handle all product data except in rather exceptional cases. It is anticipated that a systems engineering environment is optimized to serve the systems engineering team in their development effort.

**Master Product Definition Domain.** The Master Product Definition is anticipated as the unique environment holding all the information about the evolution of system releases. It maintains the information throughout the whole system architecture and covers all engineering domains. Furthermore, it controls all the data handed over to manufacturing.

All authorized users of the Master Product Definition are potential stakeholders for the systems engineering information in its repository. This demands the availability of systems engineering information that is valuable and meaningful to this wider community. The systems engineering information needs to be presented in a way understandable and interpretable by the ordinary user. Some complexity, non-avoidable in a systems engineering environment, but only relevant for the systems engineering team should not be exposed in the Master Product Definition.

## Integration of Multiple Systems Engineering Environments

**Multiple Systems Engineering Environments.** When discussions on the systems engineering integration into PLM started within the working group, it became evident immediately that we would have to deal with multiple systems engineering environments. A vision for selecting specific systems engineering tools and integrating them deeply within the Master Product Definition would be an illusion. Needs were to diverting for a number of reasons.

The expectations on systems engineering tools vary with the system life cycle phase. For example, during conceptual design the aim is to find an optimum solution for a given problem. Main criteria comprise effective mission performance and affordability in the presence of uncertainty and risks. In contrast, full consistency and completeness of the system definition is the expected outcome of a definition phase. Systems engineering tools that support the definition phase well may be a burden for the conceptual design phase where completeness is not the goal.

The wide range of products and services leads to different demands on the systems engineering environment. For some projects, the definition of an appropriate system architecture may be the main challenge. For others, the system architecture may be more or less pre-defined and functional and performance fine-tuning are the main topics to work on.

Huge variations in the duration of system life cycles are a further EADS-specific issue. Especially in the aeronautic field, legal and regulatory obligations demand the capability to maintain the fleet in airworthy conditions for the full in-service life. Experience gained by now leaves little confidence that systems engineering tools will be supported for so long by tool vendors. On the other hand, migrating the systems engineering data every time data formats or tools are changing is no real option due to economical reasons. Thus, it is a fact that various systems engineering environments for newly developed systems and legacy products have to be supported concurrently.

It is recognized that in complex industrial organizations like an extended enterprise with partner and multi-tier supplier organizations, PLM is the foundation of the cooperation (Messadia, Eynard and Sahraoui 2011). But this does not prevent from facing variations of the systems engineering environment. Due to high competition and contractual constraints every organization will seek for continuous improvements in their processes to keep or to increase their profits.

**Integration Objectives.** Figure 3 illustrates the interaction of the Master Product Definition with various systems engineering environments. For integrating the systems engineering information, the Master Product Definition provides a Systems Engineering Interface (SEI) Data Model. A consistent use of the SEI Data Model needs to be ensured independently of the original systems engineering environments by which the data is generated. Five usage modes provide guidance for filling up the data structures according to the needs of a particular program or project.

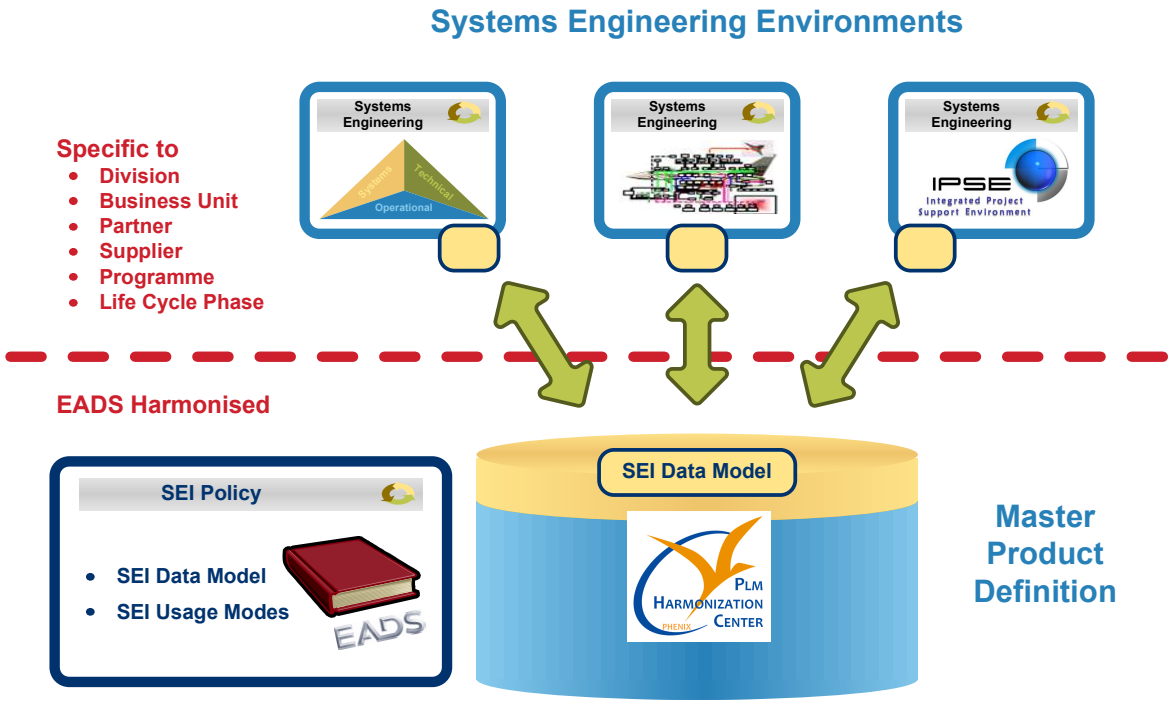


Figure 3. Integration of Multiple Systems Engineering Environments

The Systems Engineering Interface is designed to satisfy the following objectives:

- To make systems engineering data available to all authorized users of the Master Product Definition Repository including
  - those who have no access to the systems engineering environment, and
  - those who do not need to cope with the intricacies of the systems engineering environment.
- To establish relations of systems engineering data with other data stored in the Master Product Definition Repository in order to provide a global view.
- To provide efficient and comfortable audit trails over all product data stored in the Master Product Definition Repository including systems engineering data for future investigations.
- To enable an enterprise-wide archiving solution for systems engineering data independently from a particular systems engineering environment.
- To allow shutting down the original systems engineering environment used for design after system development has been completed instead of maintaining it for the development of potential future updates.
- To facilitate the import and export of systems engineering data between the Master Product Definition Repository and various current and future systems engineering environments on the basis of a common data model.

**Avoiding Synchronization Pitfalls.** The chosen architecture to integrate multiple systems engineering environments leads to a duplication of data stored in the systems engineering environment repository and the Master Product Definition Repository. Some rules need to be established in order to avoid inconsistencies within a single repository and between both repositories.

The first rule is to control the application data processing for certain data in one environment only. In a very sophisticated environment this would clearly be the systems engineering environment for all systems engineering data. Under more realistic conditions, the systems engineering environment may lack some functionality provided by the Master Product Definition. Thus, some systems engineering data may be generated or further processed by the Master Product Definition.

The second rule is to export only mature systems engineering data from the systems engineering environment to the Master Product Definition. In this instance mature means two things. The quality of the data has been evaluated. And, the data may be used as a point of reference for the work of other users of the Master Product Definition. This rule avoids inconsistent states of immature data linked to other objects in the Master Product Definition.

The third rule is closely coupled with the second. In order to enhance concurrent engineering capabilities export cycles of systems engineering data should be short. For the systems engineering activities, this means that process capabilities for managing iterations performed concurrently are available and are applied.

The fourth and final rule forbids the export of systems engineering data for which editing is controlled by the Master Product Definition to a systems engineering environment. Otherwise, inconsistent links could be created when systems engineering information is exported back to the Master Product Definition Repository. The information within the Master Product Definition may have further evolved meanwhile. However, there is one exception to this rule: A systems engineering environment repository may be initialized with data from the Master Product Definition Repository before commencing development activities in the particular systems engineering environment.

## The Systems Engineering Interface Data Model

It is the main purpose of the SEI Data Model to publish systems engineering information for the benefit of all authorized users of the Master Product Definition. As a consequence, systems engineering data may be linked to other data stored in the Master Product Definition Repository for achieving overall PLM. Although the SEI Data Model is intended to initialize systems engineering environments, it does not provide dedicated data structures for all specific features of particular systems engineering tools, e.g. of presentation attributes and storage schemes.

**Usage Modes and Their Dependencies.** Not all systems engineering efforts will populate the complete SEI Data Model with data depending on the particular needs. However, in order to ensure a consistent view within the Master Product Definition, usage modes are defined to provide guidance for which purposes and under which conditions the sub-sets of the SEI Data Model shall be used. Figure 4 shows the five usage modes and indicates their dependencies.

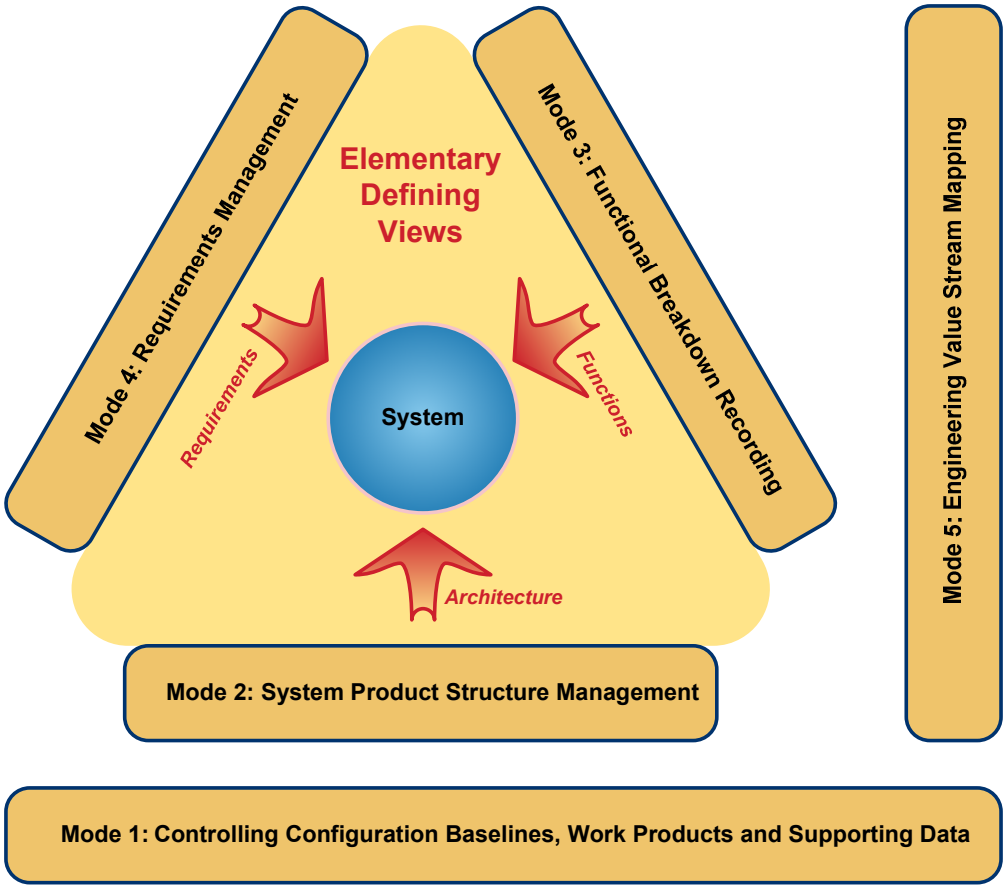


Figure 4. Usage Modes

Usage Mode 1, "*Controlling Configuration Baselines, Work Products and Supporting Data*", satisfies basic configuration management needs for compliance with legal and regulatory obligations. Its application is mandatory. All other usage modes are built on top of Usage Mode 1.

Usage Modes 2 to 4 are providing the architectural, functional, and requirement views essential for requirements engineering and system design. Usage Mode 2, "*System Product Structure Management*", is of use whenever several system elements are developed by loosely coupled, but widely independently managed value streams. It enables the control of any concurrent engineering practices between those value streams. Usage Mode 3, "*Functional Breakdown Recording*", is not intended to create a separate product structure in parallel to the system product structure. Instead it maps the result of the functional analysis for the related system element. Usage Mode 3 becomes increasingly valuable, if Usage Mode 4 is applied also. Usage Mode 4, "*Requirements Management*", adds the representation of requirements and traceability as individual entities to the SEI Data Model. Thus, detailed impact analysis over the whole system product structure and other data linked to it is enabled for requirements. In conjunction with Usage Mode 3, it becomes visible which requirements are relevant for which function and sub-function.

Usage Mode 5, "*Engineering Value Stream Mapping*", is only dependent from Usage Mode 1. Its purpose is to control the evolution of a system element when iterative development practices are applied.

**Systems Engineering Interface Data Model Class Diagram.** Figure 5 shows a class diagram depicting the SEI Data Model. In the following the object classes and their purpose are briefly described. The object classes are assigned to specific usage modes as shown by different colors used for each usage mode.

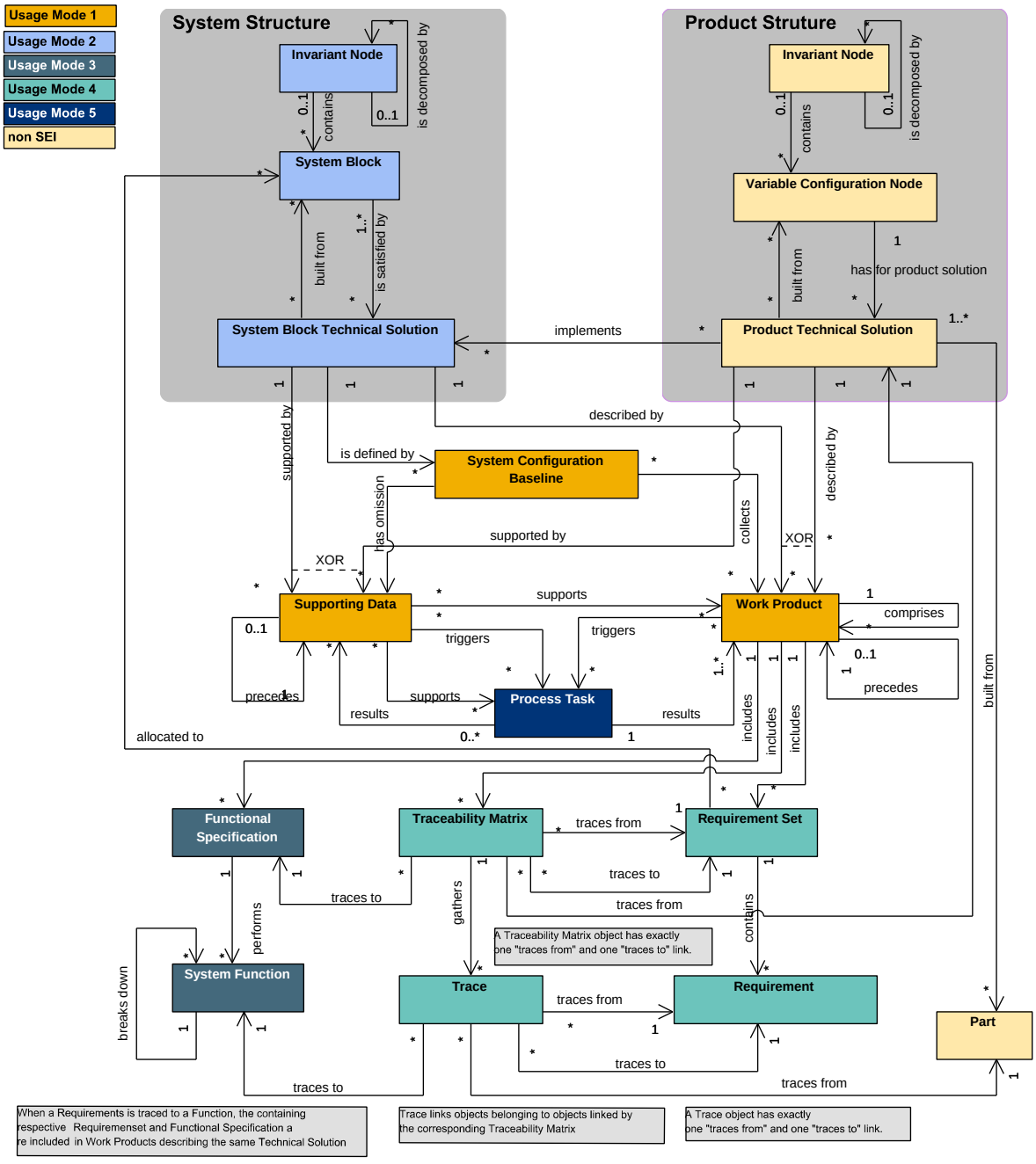


Figure 5. SEI Data Model

**Usage Mode 1: Controlling Configuration Baselines, Work Products and Supporting Data.** Three object classes are associated with Usage Mode 1: System Configuration Baselines, Work Product and Supporting Data.

Work Products contain the engineering value generated. They have to be kept up-to-date and are designated for being referenced by System Configuration Baselines. Examples for Work Products include all artifacts assumed by ISO-10007 (ISO 2003) as product configuration information like specifications, design documents etc., plus the assurance results like test reports and safety assessment results. The content of Work Products may be stored in various files with different formats ranging from readable documents to tool proprietary data formats. All artifacts that are not Work Products are called Supporting Data. They are of a more temporary nature and are valid only in the context of the Work Products they are contributing to. Examples for Supporting Data include all change control records, trade-off studies, review comments and responses, and communication records like letters, meeting minutes, and e-mails. While Work Products tell what the system is, Supporting Data provide all the hints why a system has evolved to its current state. As far as this is helpful to analyze later improvement suggestions, problems or accidents, Supporting Data should be kept. The distinction of these two categories of information was first proposed in an earlier paper (Scheithauer and Schindler 2000) for two reasons: improving the efficiency of managing life cycle data, and enabling the definition of engineering value streams (Scheithauer 2012).

System Configuration Baselines may fulfill a number of purposes in order to serve as a point of reference comprising a consistent set of several Work Products. Thus, they may support any management review for investment decisions or for assessing, if objectives of an investment have been achieved. On a lower technical scale System Configuration Baselines may for example be defined to control the test readiness of individual test procedures. Considering all the information stored in the Master Product Definition Repository, System Configuration Baselines may furthermore be utilized for continuously monitoring their evolution and the identification of the delta for achieving an associated quality gate.

**Usage Mode 2: System Product Structure Management.** The System Product Structure captures a system's architecture. The overall system and every system element on any architectural level are represented by two object classes: System Block, and System Block Technical Solution. The System Block is generated by the systems engineering team in charge of the upper level system. It stands for the requirements allocated to the system element. On the basis of the allocated requirements, the System Block Technical Solution is defined in terms of system requirements, system functions and the architectural breakdown to the system elements on the next lower level. When a System Block Technical Solution represents an item in the 3D world, it needs to be linked to the corresponding Product Technical Solution that exists outside the SEI Data Model.

As described, each system element is represented in the SEI Data Model by two objects. This enables variant management and re-use. A System Block may refer to more than one System Block Technical Solution.

A further object class, Invariant Node, may be used to map a pre-defined product structure. Such a system product structure may result from a concept phase. In the concept phase, a high-level system architecture may be defined without any detailed work commenced on the individual system elements. Another purpose may be the mapping of system breakdown codes of a standardized system breakdown. This is common practice in aviation for maintenance purposes, for example ATA Specification 100 (ATA 1999).

**Usage Mode 3: Functional Breakdown Recording.** The central object class of Usage Mode 3 is called System Function. It represents a function or sub-function. The trees of nested functions and sub-functions are intended to record the functional decomposition performed for a particular system element. The functional decomposition may go much deeper than the decomposition into system elements on the next architectural level in order to generate appropriate understanding and the evidence for the reasonability of the architectural decomposition.

A set of top level functions is assembled as a Functional Specification object. A Functional Specification is designated to be included in a Work Product. In conjunction with Usage Mode 4, traces to requirements and parts may be established.

**Usage Mode 4: Requirements Management.** Usage Mode 4 is dedicated to detailed requirements management information. Individual Requirements may be bundled to Requirement Sets that are then linked to other objects, namely Work Products, or in case of allocated requirements, to a System Block. Traceability information may either be contained in Traceability Matrices or may be expressed as explicit relations between requirements. Requirement Sets and Traceability Matrices are linked to those Work Products that represent their content. Furthermore, individual requirements may be linked to Parts that exist outside the SEI Data Model.

Usage Mode 4 offers the foundations for performing Product Lifecycle Requirement Management (Carlsson and Strandberg 2009). In conjunction with Usage Mode 3, the relations between System Functions and Requirements support the generation of functional based test cases with coincidently creating the evidence for a requirements based demonstration of compliance. In conjunction with Usage Mode 2, the relation between a system block and its allocated requirements allows to reflect the cascade of requirements and the link with the design activity, providing a way to manage the "engineering sandwich", composed of subsequent layers of models and requirements (Dick and Chard 2004).

**Usage Mode 5: Engineering Value Stream Mapping.** The single object class associated with Usage Mode 5 is named Process Task. It represents the process task level of the hierarchical process model introduced by an earlier paper (Scheithauer and Schindler 2000). Process Tasks connect all Work Products with trigger/result links to build up the work product generation sequence. To some extent also Supporting Data may be included in a work product generation sequence (Scheithauer 2012). Especially, configuration control records may be used to express the front loading of the value stream.

A Process Task generally stands for the activities performed in order to generate a particular version of a Work Product. The supports relation between the Process Task and the contributing Supporting Data defines the validity context of the Supporting Data.

## Conclusion

The considerations presented in this paper lead to a sustainable as well as versatile solution for integrating systems engineering into an overall Product Lifecycle Management solution. The main benefits of the proposed approach lie in four areas.

At first, the Master Product Definition holds all the systems engineering information to fulfill the stated objectives. All systems engineering data and the links to other product data are traceable using standard features of the Master Product Definition. No additional tools are required in principal. However, powerful data representation applications will usually serve comprehensive read-only views supporting the interpretation of the data stored in the Master Product Definition repository.

At second, the data model of the systems engineering interface sets a sustainable standard for representing the systems engineering information. It is comparably simple. Authorized users of the Master Product Definition that are not systems engineering specialists have not to bother with the intricacies of a systems engineering authoring environment.

At third, flexibility is gained by a modular concept in two directions. Usage modes are to be selected according to actual program rules or project needs with an impact on the actual capabilities for information tracing in the Master Product Definition. In the other direction, a further evolution of systems engineering methods and corresponding suitable systems engineering environments is not constrained. Changes to the systems engineering environment have no impact on the data model of the systems engineering interface or the stored systems engineering data in the Master Product Definition.

At fourth, the modular concept allows also the coexistence of several systems engineering environments connected to a single Master Product Definition. Legacy programs may stick to their traditional systems engineering environments while emerging programs may select a high sophisticated systems engineering environment applying leading edge methods and tools. In the context of complex organizational set-ups like a supply chain, some degrees of freedom are gained for all the systems engineering teams in an extended enterprise. When traceability over the whole system architecture is maintained by a single Master Product Definition, suppliers may be allowed to rely on their in-house systems engineering capabilities and systems engineering environment.

Despite all the listed benefits, it finally should be noted that the approach presented does not define a terminal state for the inclusion of systems engineering into Product Lifecycle Management. It is rather the starting point for further integration of the classical product data management domain into an overall consistent and efficient systems engineering process.

## References


ATA (Air Transport Association of America) 1999. *ATA Specification 100 - Specification for Manufacturers' Technical Data.*

Carlsson U., Strandberg T. 2009. "Product Lifecycle Requirement Management (PLRM) – a through life information management challenge." Paper presented at the annual PDT Europe conference, Versailles, France, 18-19 November.

Crnkovic I., Asklund U., Persson Dahlqvist A. 2003. *Implementing and integrating product data management and software configuration management.* Norwood, MA (US) Artech House.

Dick J., Chard D. 2004. "The Systems Engineering Sandwich: Combining requirements, models and design." White paper, Telelogic.

ISO (International Organisation for Standardisation) 2003. ISO 10007. *Guidelines for configuration management.*

——— 2010. ISO 10303-233. *Automation systems and integration - Product data representation and exchange – Application Protocol 233 - Systems engineering.*

Messadia M., Eynard B., Sahraoui A.E.K 2011. "PLM as a tool for supporting industry collaboration." Report N° 11339, LAAS Toulouse, France.

Mondon, J.Y. 2009. "PHENIX - The EADS Programme for PLM Processes, Methods & Tools Harmonization." Paper presented at the annual PDT Europe conference, Versailles, France, 18-19 November.

OMG (Object Management Group) 2011. *ReqIF - Requirements Interchange Format.*

Scheithauer D., Schindler A. 2000. "A Standardisation Concept for Non-Standard Development Projects." Paper presented at the EuSEC 2000, Munich, Germany, September 13th – 15th.

Scheithauer, D. 2012. "Managing Concurrency in Systems Engineering." Paper presented at the 22nd INCOSE International Symposium, Rome (IT): 9-12 July.

# Biography

**Frédéric Autran** is Systems Engineering Senior Expert in Cassidian (an EADS company). He has an engineer degree from Ecole Centrale de Paris (1984). After 5 years in development of CASE tools, he acted for 8 years as a consultant for the French Ministry of Defense and contributed to the building of the semantic interoperability framework for the various French Army C3I systems. He then joined EADS in 1997 to set up the management of interoperability among systems composing the new command and control system for the French Air Force, and introduced the System Engineering principles of ANSI/EIA632 in this programme. Since 2000, he is involved in the deployment of systems engineering in EADS, from process definition to project coaching. In the frame of the PLM Harmonization Center, he currently leads the "PLM4SE" group harmonization projects that defines the interface between Systems Engineering activities and the Master Product Definition. In 2005, he created and chaired until beginning of 2009 the System of Systems and Complex Systems working group of AFIS (French chapter of INCOSE). He is INCOSE Certified Systems Engineering Professional since August 2009.

**Dieter Scheithauer** studied electrical engineering with special emphasis on automatic control at the Universität der Bundeswehr München resulting in the degree of a Diplom-Ingenieur univ. in 1980 and a doctor's degree (Dr.-Ing.) in 1987. His service as technical officer in the German Air Force ended in 1988. Over his professional career he contributed in various roles to the flight control system development for major European military aircraft and helicopter programs. He acted as project manager for the development of unconventional airborne and ground-based systems. In 1999 he joined the European Aeronautic Defence and Space Company. Since then he has mainly worked in the field of process engineering. Today he holds a position as Senior Expert Systems Engineering Processes within Cassidian.
He is a former president and a honorable member of GfSE – The German Chapter of INCOSE. He represents Cassidian on the INCOSE Corporate Advisory Board. And, he is an INCOSE Expert Systems Engineering Professional.