

Eurostep Technology White Paper A Shared Data Environment for Requirements v1.0

Ian Bailey

Executive Summary

This white paper describes some initial ideas about a capability for managing requirements across a number of organizations using different requirements management tools. The main topic discussed is the need to trace between requirements that originate in different parts of an organization or supply chain, and may have been authored using different tools. The relationship between requirements and PDM data is also considered. Finally, the *big picture* of a fully integrated "Lifecycle SDE" is discussed. It is intended that this document will evolve over time as the ideas are implemented.

The business of engineering and manufacturing complex products is highly distributed. Supply chains are tightly integrated and are usually able to share and exchange information electronically. Unfortunately, the customer (or even the OEM) tends to be less integrated – they often just issue the requirements, approve the proposals and take ownership of the product (and a bundle of paper and electronic support documents). Obviously, this is not the optimal situation.

The idea of a *lifecycle SDE* enables a level of integration that up until now has not been possible. The original requirements can be traced back from any stage in the lifecycle – design, manufacture, installation, operation, maintenance, and in-service modifications. This allows the systems engineers to verify multiple sources of requirements against emerging system properties. It also allows design trade-off decisions to be captured in perspective of the whole product lifecycle.

In the operations phase, a *lifecycle SDE* offers the ability to provide design information to maintenance engineers that can be kept up to date with through-life modifications. It also allows operators to check if failures cause the product to fail to meet its original requirements (e.g. safety requirements). In addition, ad-hoc repairs and modifications can also be referred back to requirements. Having the complete product lifecycle picture offers endless possibilities for increased efficiency.

Finally, the paper discusses how the Share-A-space™ SDE is being extended to become a true lifecycle SDE.

Introduction

The modern engineering process is a collaborative effort - supply chains are evolving into collaborative teams, enabled by advances in IT. Shared data environments (such as Share-A-space™) enable engineers to securely and selectively share configuration-managed product structures. These sharing systems allow change control and notification across an extended enterprise and enable more efficient engineering design collaboration. It is becoming common practice to share CAD and PDM data electronically.

The business of managing and sharing requirements across enterprises is less well understood, and has not been attempted on the scale of projects which share PDM and CAD data. The current approach to sharing requirements is to send documents between companies. This leads to problems of poor traceability, slow update times and errors which stem from manual re-entry of data. Even within the same company, it is not unusual for two departments to employ different requirements management tools. In these situations it is difficult for changes in one set of requirements to be registered in the other set.

Although the importance of requirements in the product lifecycle is recognized, it is rare for requirements to be integrated with other engineering information. In particular, it is important to capture the design decisions (e.g. trade studies) that result in designs which satisfy the requirements and the various models (system, functional, physical) which are used when allocating requirements. This is often called "requirements traceability". Beyond this level of traceability, there is also interest from the owner-operators of large systems who need the ability to trace back from operating products to the original design requirements. This enables users to validate any in-service modifications against the original requirements for the product.



There is an international standard for systems engineering and requirements data - ISO-10303-233 (known as AP233) and for product lifecycle support (PLCS) data - ISO 10303-239 (known as AP239). These two standards have cooperated in their development and are compatible in areas where they overlap. The purpose of this paper is to explain how these standards can be implemented in a data sharing environment to enable a level of traceability that has not been possible until now. The factors which govern good shared data environments apply to requirements management and sharing:

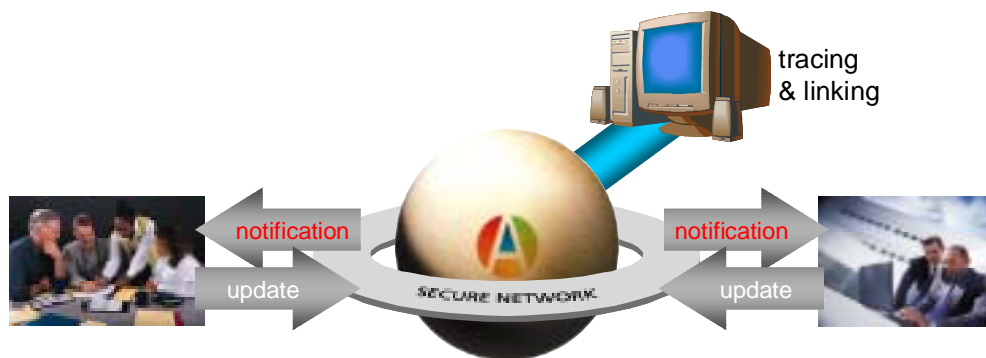
- Users are able to use their preferred software tools for originating and viewing data
- Secure, web-enabled user-interfaces for those users who do not have software tools
- Ability to keep own business processes in place
- Provision of change notification and workflow integration
- Selective publication of information to different parties for IPR and security reasons
- Control over where the master records reside
- Use of standard data formats and industry-standard technologies

Sharing Requirements

The simplest way to share requirements is to publish them (in a secure way) to project partners. The owner of the requirements may wish to select which project partners have access to the requirements, and allow the partners to issue comments and feedback through the sharing system. Requirements usually originate in a software tool (e.g. a requirements tool, or simply a word processor) and can be uploaded to the shared data environment (SDE). Changes in the original requirements may also be uploaded, and the SDE should be able to trace these changes – i.e. version history. It is not always convenient to update the data to the SDE as and when changes are made – hence the SDE should provide continuous *and* batch update capabilities.



The more advanced case for sharing is when the requirements engineering process is distributed between teams. This may involve an engineering contractor developing system requirements which are derived from original customer requirements. Alternatively, it may be a company (or supply chain) where individuals or departments are responsible for different sets of requirements which may have impact on each other. It is likely that the requirements are managed in different tools, and it is important to link change control with traceability between the different requirement sets. For example, it is not uncommon to find that the systems engineers use DOORS™ (from Telelogic) while the software engineers use Requisite Pro™ (from IBM). When requirements in one tool change, these must be detected and notifications sent out to the affected parties.



As the requirements have originated in different places, it is unlikely that tracing and linking information will be available between the two requirements sets. Hence, the SDE should provide a user interface for establishing the links between the two sets of requirements.

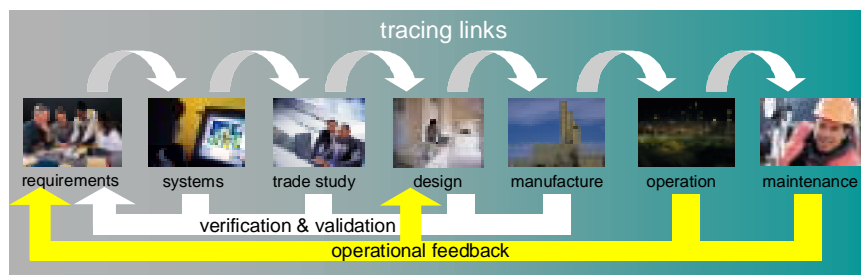
Requirements can originate in many different forms – most often as a requirements document. In this case, the individual requirements are usually represented as sections or tables in the document. The requirements are generally specified as text, but may also be specified as graphs, drawings, models, equations, numeric bounds or tables of values. Any shared data environment for requirements should be capable of handling any of these representation types. It is preferable for the SDE to manage each requirement as a data element instead of managing the document as a “dumb” object. However if it is not possible to extract the individual requirements from the document in a computer-sensible manner, it may be necessary to manage data on a document level.

Organizations are increasingly using behavioural models to specify requirements. In the defence industry this is known as “simulation based acquisition” and is receiving a lot of attention at the moment, but the idea of using behavioural models is not new to systems engineers. The question for SDE implementers is how to handle behavioural models. As a minimum, the SDE should be capable of storing the behavioural models as files (in the format used by the originating tool). It should be possible to configuration-manage these files and link them to text requirements, and system/physical/functional breakdowns. However, to fully enable collaboration with behavioural data, the SDE should be able to manage the individual data elements. Although this is not a simple undertaking (the SDE will have to integrate closely with the model authoring tools), the SysML standard may well offer the necessary structures for managing this data at a fine-grain level.

Requirements in a Lifecycle SDE

As well as extending the SDE functionality to early lifecycle information (i.e. requirements), there is strong demand for sharing information about the operational phase of a product which is integrated with the design and requirements information. What we will call here a “lifecycle SDE” integrates requirements, systems, trade-off, and design data with operational data about individual products – usage, maintenance, re-fit, re-design, etc.

The lifecycle SDE is capable of handling data which originates in a variety of software systems. It is also capable of establishing the links between data elements from different phases of the product lifecycle, and capturing the justification for those links. For example, it may contain a set of original user requirements which have been mapped onto a set of system requirements. The lifecycle SDE would capture the tracing relationships between the individual requirements in each set. It would also capture the various systems engineering breakdowns which may have been used (e.g. functional, physical, system) and how the requirements trace to the individual elements in those breakdowns. The SDE may also be able to capture functional modeling information and provide tracings to the appropriate breakdowns and requirements.



If required, the lifecycle SDE should also capture information about design trade-off – i.e. how a particular design was selected for a system. It is also important to consider the origin of the requirements – including when new requirements result from a design decision. All this information is then linked into the PDM data (i.e. product structure, version control, etc.). The SDE should provide a “requirements redlining” capability so that comments can be added to requirements specifications. In addition, as properties start to emerge from design and analysis, these can be verified against the requirements. As prototypes are manufactured, and systems put into operation, the validation of requirements can be tracked in the shared data environment.

The final part of the lifecycle SDE is the capability to track products whilst they are “in service” – i.e. serial numbered items. This may also include maintenance information or close integration with an existing maintenance management system. The lifecycle SDE provides the links from the in-service items to their design information (PDM). It also provides the feedback mechanism for in-service fault reporting.



By sharing all this information, a lifecycle SDE provides complete lifecycle traceability, but still allows the master data to remain in the originating systems. In other words, the SDE acts as a “hub” for the data and synchronises information between the different originating software systems. It is not uncommon for an SDE to offer change notification – users subscribe to be informed if a particular item changes. However, a lifecycle SDE has to offer an additional level of change notification – users are notified if there is a change to an item which is traced to or from the item they are interested in. For example, the owner of a design should be notified if one of its driving requirements is changed. Similarly, the maintenance team for a particular in-service item should be notified of a design change.

In developing any shared data environment, it is important to use data standards which do not lock the user into any proprietary tools or formats. Of particular interest are two ISO standards:

- ISO10303-233 (AP233) – the standard for systems engineering data exchange. This emerging standard specifies data structures for requirements, systems, behaviour, verification and validation, analysis, etc. It also provides the traceability from requirements through systems and functional models to design trade studies.
- ISO10303-239 (AP239) – the standard for product lifecycle support (PLCS). This standard defines structures for exchanging information about operations and maintenance of engineering products and provides links from design information to operational information.

In addition, the SysML standard (INCOSE-led extension of UML for systems engineering) provides an excellent basis for communicating the diagrammatic aspects of the shared systems engineering data. The DoD Architecture Framework also provides a useful approach for categorising systems engineering models, and any SDE should consider using DoDAF as a basis for sectioning and displaying the data.

These standards, if used in conjunction, offer the possibility to manage, share and exchange information throughout the product lifecycle. Because the standards are well documented and vendor-independent, implementations based on these standards will not “lock in” data. In a shared data environment, the standards can be used in two ways; to define the internal structure of the shared data repository, and to provide a neutral file format for importing and exporting data.

Way Forward

The technology required to implement a lifecycle SDE already exists in the form of the Share-A-space™ environment. Share-A-space™ is a software product which enables sharing of complex engineering data. Collaborating companies can use Share-A-space™ to selectively and securely share appropriate parts of their engineering data. Access is strictly controlled, and users only share the information they want to share – i.e. the partners do not have access to all the data in each others’ CAE systems. Share-A-space™ also enables smaller partners, who may not have their own PLM system to work in a collaborative PLM environment (all they need is a web browser and an internet/extranet connection).

Share-A-space™ is based on ISO standards for product data (PDM Schema and PLCS), and allows sharing of core PDM data (product structures, configuration control, properties, documents, etc.). The product has recently been extended with aspects of AP239 to enable through-life product support information to be shared (in-service configuration, operational feedback, maintenance, etc.).

The next phase for Share-A-space™ is to add a requirements sharing capability and to integrate this with the existing product data capabilities. Clearly, there is a lot more to systems engineering than just requirements, and other aspects such as trade studies, V&V, behavioural models, risk, etc. will also be implemented in Share-A-space™ as and when customer demand dictates.

Eurostep has already developed a number of prototype AP233 interfaces to commercial requirements management and systems engineering tools (DOORS™, Requisite Pro™, etc.). In addition, vendors have also implemented AP233 interfaces to their tools (EDS Slate™, 3SL Cradle™). Share-A-space will act as a hub for integrating data (using AP233) from these sources, as well as data from PDM, CAD, manufacturing and maintenance systems.

Contact

e-mail	ian.bailey@eurostep.com
web	http://www.share-a-space.com
	http://ap233.eurostep.com