ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands

T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

IDEALISM
Integrated & Distributed Engineering Services Framework for MDO

D4.1.3: Requirement Specification and Standards for the Engineering Language Workbench - final

Author, company:
Marc Eheim, IILS
Stephan Rudolph, University of Stuttgart
Roland Weil, IILS
Jürgen Freund, University of Stuttgart
Kjell Bengtsson, Jotne
Maarten Nelissen, KE-works
Erwin Moerland, DLR
Roberto d'Ippolito, NOESIS
Martin Motzer, DRÄXLMAIER
Kevin van Hoogdalem, KE-works
Jochen Haenisch, Jotne
Franz Stöckl, DRÄXLMAIER
Til Hendrich, KE-works
Akshay Raju Kulkarni, TU-Delft
Marco Panzeri, NOESIS
Sebastian Deinert, Airbus Defence & Space
Maurice Hoogreef, TU-Delft
Johnny van Lugtenburg, Fokker Elmo

Version:
1.2

Date:
October 12, 2017

Status:
Final / Released

Confidentiality:
Public

CHANGE LOG

Vers.	Date	Author	Description
0.1	29.05.2017	Marc Eheim	- Removed not used requirements for the use-cases based on V&V D5.1.2; put obsolete requirements into the annex for traceability; - Modified chapters 1 and 8 for final version; - added one requirement to cover user stories (Req-ELW-CH-5)
0.2	30.05.2017	Marc Eheim	updated tool description for DC43 (in chapter 5)
0.3	30.05.2017	Marc Eheim	Some minor modifications in chapters 2 and 6
0.4	01.06.2017	Marco Panzeri	Updated tool description
0.5	06.06.2017	Marc Eheim	status: To be reviewed
0.6	20.06.2017	Johnny van Lugtenburg	Review performed.
0.7	23.06.2017	Marc Eheim	Processed reviewers comments; Added commonly used file formats; re-added GBDL physics requirements; added one additional requirement;
0.8	01.08.2017	Sebastian Deinert	Added requirement in section 4.2.1
0.9	11.08.2017	Erwin Moerland	Added requirements in section 4.2.1
1.0	14.08.2017	Marc Eheim	Merged document with requirements for HDOT, finalised
1.1	21.08.2017	Marc Eheim	Removed requirements of engineering services
1.2	12.10.2017	Johnny van Lugtenburg	Added description of the Engineering Services in scope of UC2 in section 4.2.1.

Table of Contents

List of Abbreviations

AIF	Advanced Integration Framework
AP	Application Protocol
API	Application Programming Interface
BPMN	Business Process Model and Notation
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CFD	Computational Fluid Dynamics
COTS	Commercial Off-The-Shelf
CPACS	The Common Parametric Aircraft Configuration Schema
CR2	Change Request 2
DSL	Domain Specific Language
EL	Engineering Library
ELW	Engineering Language Workbench
EWIS	Electrical Wiring Interconnection System
FEM	Finite Element Method
FPP	Full Project Proposal
GBDL	Graph-Based Design Language
HLDL	High Level Design Language
IGES	Initial Graphics Exchange Specification
JT	Jupiter Tessellation
KBL	Kabelbaumliste
MDM	Master Data Management
MDO	Multi-disciplinary Design Optimization

MoSCoW	Must haves, Should haves, Could haves, Won't haves, according to MoSCoW prioritization technique (see Annex C: Requirements classification)
OEM	Original Equipment Manufacturer
OWL	Web Ontology Language
PDM	Product Data Management
PLM	Product Lifecycle Management
RDE	Resource Description Framework
STEP	STandard for the Exchange of Product model data
SysML	Systems Modeling Language
UML	Unified Modeling Language
VDA	Verband der Automobilindustrie
VDAFS	Verband der Automobilindustrie - Flächenschnittstelle
VEC	Vehicle Electric Container
VHDL	Very High Speed Integrated Circuit Hardware Description Language
W3C	World Wide Web Consortium
WP	Work Package
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

1 Introduction

The objective of Work Package 4 (WP4) is to develop an Engineering Language Workbench (ELW), providing a service development toolkit to generate flexible engineering workflows and services featuring a set of domain specific and high-level modelling languages, ontologies and standard interfaces and data formats. It enables the generation and integration of engineering services and workflows into the Advanced Integration Framework[1] (AIF), developed in WP3 of the IDEaliSM project. Besides describing used data standards, the purpose of this deliverable is to specify the functional- and technical requirements for each of the sub-components of the ELW. The requirements of the Engineering Library (EL) are described as well, since the EL is a task of WP4. Following the MoSCoW (see Annex C: Requirements classification) method, the requirements were initially based on the FPP [1], revised for the new tasks added with change request CR2 [2] and were more detailed with the Use Case Specifications D2.1.x in three iterations (see [3], [4] and [5]).

WP4 follows the overall iterative approach in IDEaliSM and thereby delivers three versions of the ELW during the project. During each of the iterations, the sub-components of the workbench are matured, seamlessly serving the building phase of the use cases within the project.

This is the third and final iteration of the deliverable for requirement specification and standards for the ELW (D4.1.3). The final requirements analysis is based on the second version of validation and verification of the demonstrators [6] of WP5 and the final definition of the Use Case Specifications [5] of WP2. It forms the foundation to establish the final iteration of the ELW and the EL.

The ELW is based on the development and use of High Level Design Languages (HLDLs) to enable the engineers to create automated engineering services. For the interoperability of the engineering services in the AIF, a common knowledge base is needed as well as standardized interfaces and data formats such as CPACS, STEP and UML.

This document is organised as follows:

- Chapter 2 contains a description of the state-of-the-art of data exchange standards (section 2.1) and the vision of the standardisation strategy within IDEaliSM (section 2.2)
- Chapter 3 describes the requirements of the ELW and its contents
- Chapter 4 describes the requirements of the EL and its contents
- Chapter 5 provides an overview of the current state of used tools in the consortium and their supported data formats and standards
- Chapter 6 contains a description of the Standard Interfaces and Data Formats which play a key role in the project setup
- Chapter 7 contains wide-spread data formats, which are used in design processes every day
- Chapter 8 concludes the most important aspects

[1] Requirements of the Advanced Integration Framework are described in D3.1.3 [1].

2 Engineering Language Workbench

The Engineering Language Workbench (ELW) serves the ultimate goal of flexibly creating engineering services and workflows for multi-disciplinary simulation and analysis models, as well as tasks and optimizations. It therefore heavily relies on a set of High-Level Design Languages (HLDLs) and domain specific languages (DSLs) with their ontologies. Together with data standards, these languages enable a flexible configuration of engineering workflows and services and a straightforward integration into the distributed Advanced Integration Framework (AIF). Sections 2.1 describes the current state-of-the-art concerning interfaces and exchange formats before section 2.2 describes the IDEaliSM vision for achieving the project goals.

2.1 State-of-the-Art

The design of complex engineering products and systems involves the concurrent development of hardware and software. Furthermore, the current design and development processes for engineering complex systems as reflected in the project use cases (aircraft design, 1-month rudder, 3-weeks cockpit and 10-day harness) are characterized by heavy coupling across disciplines. The design of such complex systems involves a multitude of domain specialists and typically follows a system-of-systems approach.

This system-of-systems approach starts with topology decisions (i.e. the architectural design decisions). The second step involves the dimensioning of the design parameters. Since many disciplinary models are used for disciplinary analyses, the consistency between these models in the automated model generation process plays a crucial role for a successful automation of the model generation process, frequently occurring in iterative design processes.

It is current state-of-the-art that these development steps of engineering information occur in process chains between different programs and models using a multitude of interfaces. These interfaces frequently rely on more or less well elaborated and established standards; some of these are open-source, some of these are proprietary. It is hereby a common experience that despite the fact that interfaces between major engineering modelling and analysis programs exist, a complete and consistent flow of information from one program to the other is not always guaranteed. Instead, parts of the information might be lost or distorted during the transmission over the interface. Manual rework is therefore frequently necessary to check, repair or complete an already completed digital model once it has been written by one system and been loaded into another system.

Standards are important to support collaboration. When specified appropriately, standards provide an appropriate trade-off between restriction and guidance. Today’s industry standards, like STEP, frequently date back more than 20 years. However, they are still largely underused, either because they are not always flexible or expressive enough for the specific user needs, because they are too complex and cumbersome to adhere to, because they are replaced by proprietary data and information exchange formats, or simply because they are ignored. However, standardisation in terms of information representation format is critical due to several reasons. First, data formats need to support projects during their entire life-time. In the aerospace industry this implies a life-span of 50 years or more, in order to ensure maintenance and certification issues, just to name the most important ones. Then, standards are essential for collecting, structuring, encoding and debugging engineering knowledge that is too valuable to be encoded into any form of a digital proprietary format. If that software supplier goes out of business, substantial investments on the customer side are at stake.

Interface and design information representation standards are currently controversially discussed in the automotive- and aerospace industry. This can be concluded from the long list of alternative standards such as UML, SysML, AutomationML, STEP, VHDL and all kinds of XML implementations, which have already consumed much development effort and have seen many updates. All of the aforementioned standards have both strong- and weak points (typically a standard well suited for representing geometry is not well adapted for representing functional behaviour and vice versa). Therefore, none of these standards was able to dominate and become the de-facto market standard so far.

Concerning data exchange in the automotive industry, the landscape of standards is even more heterogeneous: for the exchange of product geometry, IGES, VDAFS and more recently STEP AP242 and JT seem to become a de-facto standard, however many OEMs still insist/prefer exchanging native CAD formats in order to avoid losing (fully or in part) the internal construction logic or other relevant product data during the translation process. In the domain of electrical wire harness development a similar radical transformation process occurs as product geometry has undergone over the last 30 years in CAD systems, but in much less time. As a consequence, current standards for harness information such as the VEC (Vehicle Electric Container)[2] as the successor of the KBL[3] standard have not yet fully converged and thus undergo steady improvements. The German Association of the Automotive Industry (VDA) recommends the VEC for the exchange of harness design data across process steps.

All-in-all, it can be concluded that standards are potentially valuable, but they currently suffer from certain drawbacks (e.g. limitation in applicability) that limit them in the fulfilment of their potential. One of the possible solutions to overcome these limitations would be the development of a consistent and unified theory of design from which the needs for a finalized version of an exchange standard could be theoretically derived. By means of such a unified theory of design, it could be concluded what the real need of the information flow between different computer programs looks like, facilitating the design of an enduring standard which would be complete and consistent and therefore a worthwhile financial investment into valid and secure digital engineering process chains. However, such a unified theory of design is so far still unknown.

2.2 Vision

In the current IDEALISM project, the aforementioned deficiencies of the data exchange formats underlying the digital process chains have raised the need for the successful development of a framework consistently supporting the product life-cycle needs of addressing, manipulating and evaluating design as well as manufacturing knowledge along the entire product lifecycle.

Novel means to represent the design- and manufacturing knowledge needs to be developed in order to fully automate, semi-automatically or interactively assist such design- and manufacturing development activities and processes along the product development process. To relieve the design engineering teams by automatic model generation from tedious routine works, automated

[2] http://ecad-wiki.prostep.org/doku.php?id=specifications:vec:start
[3] http://ecad-wiki.prostep.org/doku.php?id=specifications:kbl

engineering services and workflows allow topological and parametrical product variations by re-use of design rules and design knowledge. To capture the design knowledge, High Level Design Languages are used, which can handle and capture different domain specific ontologies. With an ELW such High Level Design Languages can be developed and used to create manifold engineering services and workflows. The ELW is based on the representation of both globally generic engineering background knowledge and locally specific engineering product design and manufacturing knowledge in a re-useable engineering ontology. The ELW therefore enables:

- Representing engineering knowledge in a human-readable and digitally processable way according to the philosophical approach “design as a language” [7]
- Decomposition and structuring of the engineering design knowledge in the form of an abstract domain specific language
- Allowing the merging, mapping and extension of the knowledge representation by processing mechanisms ensuring consistency and correctness
- Model generation of all necessary disciplinary engineering analysis models by generation of consistent, domain-specific model representations

For this purpose, the concepts of High Level Design Languages will be used and partially extended. The creation of such a generic applicable language that can be used for domain specific knowledge representation involves the cooperation of several specialists, as a consequence several means have to be developed in order to ensure the capability of cooperation of specialists separated in space and time (i.e. support of concurrent distributed engineering concepts) and to automatically merge and integrate their partial ontologies into a globally consistent and system-wide accessible and valid re-useable knowledge representation.

The first goal of the ELW involves the following list of syntactical features definitions and developments:

- Demonstration of merging and integration capabilities of separated, partial ontologies into an overall, system-wide valid ontology to ensure global consistency of engineering concepts. This includes the development of consistency checks for validation and verification and the development of knowledge representation regulations to ensure the correctness of both global representation and processing during its construction.
- Demonstration of mapping capabilities of partial ontologies from one representation format (such as UML) into other data formats (such as CPACS) by means of import and/or export filters. This is tested in a first step by mapping ontology information between equivalent vocabulary and rule content represented in CPACS/STEP and UML.
- Investigation and exploration of round-trip engineering capabilities. That means an ability of establishing a potentially permanent and interactive mapping between a domain-specific language (edited in its domain-specific editor) and the generic knowledge representation in the high level design language and/or ontology.
- Interface and integration of design optimization loops via generic/abstract optimization “adaptors” coupling the design language components to the optimizer capability. These depend on the mathematical properties of the representation space (discrete decisions for topology-based methods versus parametric decisions for gradient-based methods).

The second development goal of the ELW involves the following list of semantical feature definitions and developments:

- Abstract geometry ontology representation: this involves the definition of geometry representation (for example in a graph-based design language) including the demonstration of mappings (i.e. translation capabilities) of abstract geometry elements to distinct domain-specific geometry representations in distinct domain-specific languages. This includes demonstration of extension capabilities for new geometry features. These features allow to create design trades where function is traded versus form (“form follows function”) and its inverse trade (“function follows form”), reflecting frequently occurring “top-down” and “bottom-up” design activities.
- Together with an abstract geometry, an abstract way of representing geometrical constraints will be developed. It allows the positioning of geometry components with respect to each other (i.e. component A is located “on top of” component B, or, line A “is perpendicular to” plane B, etc.).
- Validation of correct geometry constructions by checking the water-proof (continuous) property of the geometry in an automated meshing tool.
- Verification of correct geometry construction by means of dedicated test grammars which systematically test the defined design language features.

Besides the aforementioned aspects of a so-called “abstract geometry”, means to also define physical properties of objects or processes are provided. For this, an abstract physics ontology representation needs to be developed. This involves several definitions as follows:

- Physics properties (e.g. material values) have to be represented and mapped to different target systems. Demonstration and extension capability of an abstract physics ontology representation.
- Together with an abstract geometry, an abstract way of representing physical boundary conditions (e.g. flow speed at the wall is zero) is to be developed. It allows the expression of physical properties related to abstract geometry (i.e. force F “is perpendicular to” plane C, or, force F “is aligned with” line D.).
- Propagation of the physical properties and enrichment of an automatically generated mesh with these boundary conditions in an appropriate domain-specific representation suited for engineering analysis and simulation such as finite element (FEM for structural mechanics analysis)
- Validation of mesh enrichment with physical properties in a FEM-analysis process by analysing and comparing the generated simulation results with known reference cases from industry within the provided use cases.
- Verification of correct mesh enrichment with physical properties by analysing and comparing the generated simulation results of the FEM-analysis with known reference analytical results.

For the listed development goals, IDEaliSM will make use of open, internationally standardized knowledge representation standards, such as Graph-Based Design Languages based on UML, STEP (and any other data format which can be generated therefrom). This is considered mandatory for the establishment of a secure and long-term knowledge processing effort. On the other hand, IDEaliSM will critically look at the issues faced by present standards and provide suggestions for improvement (e.g. by providing proposals for future standards like CPACS). For example, most of the CAD/CAE systems are able to import/export STEP files; however, a lot of the product information and data structuring is often ignored by these systems, which severely limits tools interoperability. IDEaliSM will look at STEP standards not only to exchange product model information including CAD, CAE and PLM data, but also for the definition of the product structure ontology (STEP ISO 10303) as well as for the structuring requirements (STEP ISO 10303-209/233/-239/242).

Figure 1 shows the components of the ELW and EL with their relations as defined and described in the Engineering Language Workbench documentation (see D4.3.1 [8], D4.3.2 [9]). In the next two chapters the requirements for the Engineering Language Workbench, the Engineering Library and their components are described.

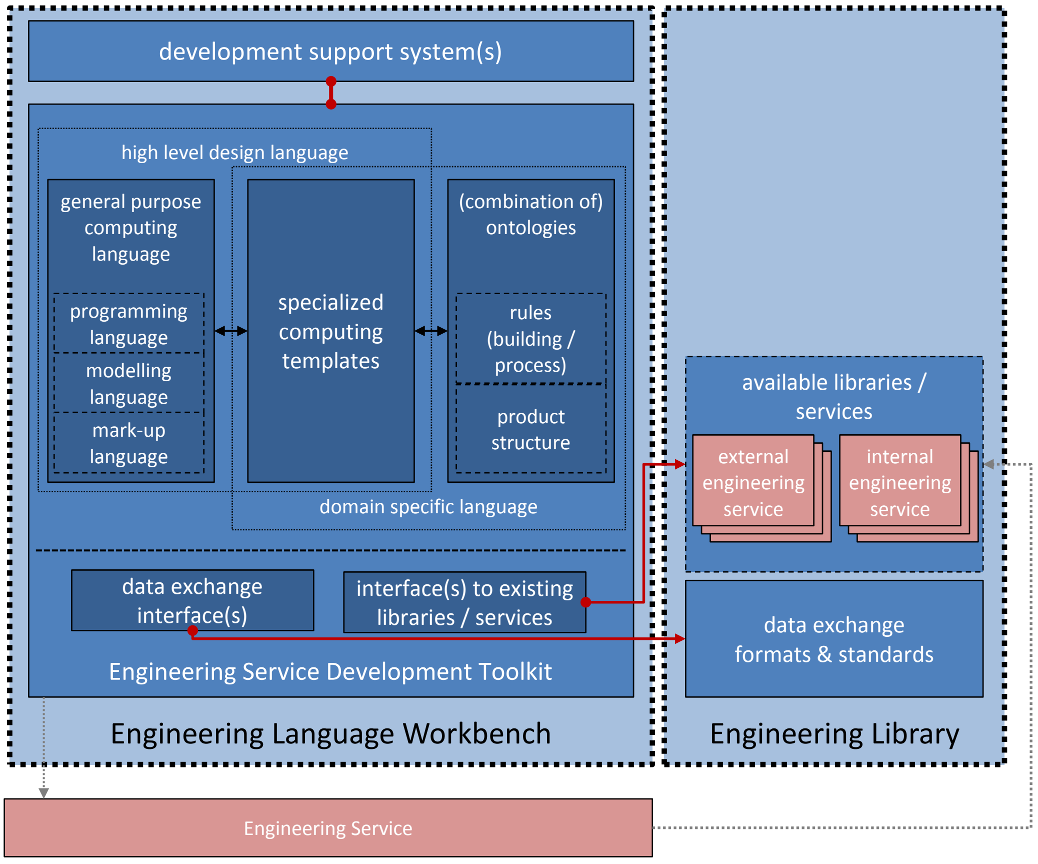

development support system(s)
high level design language
general purpose computing language
programming language
modelling language
mark-up language
specialized computing templates
(combination of) ontologies
rules (building / process)
product structure
domain specific language
data exchange interface(s)
interface(s) to existing libraries / services
Engineering Service Development Toolkit
Engineering Language Workbench
available libraries / services
external engineering service
internal engineering service
data exchange formats & standards
Engineering Library
Engineering Service

Figure 1: Components of the ELW and EL with their relations

3 Requirements for the Engineering Language Workbench

3.1 Engineering Language Workbench

3.1.1 Description

The creation of an Engineering Language Workbench is necessary for the application of High Level Design Languages. This will be accomplished if methods could be defined and applied for decomposing and structuring engineering design knowledge. These High Level Design Languages (requirements described in section 3.2) can be used to create various engineering services and workflows. They are a great tool for packing engineering knowledge into a formalized representation. But to complete such languages, efficient processing mechanisms for merging, mapping and extending the knowledge have to be developed ensuring consistency and correctness at all times.

This involves the following developments:

- Automated merging and integration capabilities of partial ontologies into an overall ontology
- Establishment of round-trip engineering capabilities between a domain-specific language (edited in a domain-specific editor) and the design language and/or ontology
- Interface and integration of design optimization loop via generic workflow “adaptors”.

3.1.2 Component requirements

Table 1 is a summary of the high-level requirements derived from the discussions above. For the requirements classification the MoSCoW method is used (see Annex C: Requirements classification).

Table 1: Requirements of the Engineering Language Workbench

Area	(Identifier) Requirement	Description	Classification
Functional	**(Req-ELW-DLW-1)** Tool Creation	Ability to develop automation tools that can create and adapt product models, including CAD (D2.1.1)	MUST
Functional	**(Req-ELW-DLW-2)** Represent knowledge	Enable representing engineering knowledge in a (human-readable and) digital machine-executable way (FPP)	MUST
Functional	**(Req-ELW-DLW-3)** Decomposition and structuring knowledge	Decomposition and structuring of the formalised engineering design knowledge in the form of design languages (FPP)	MUST
Functional	**(Req-ELW-DLW-4)** Rule based modification	Knowledge rules must be (re)configurable in standard libraries, to achieve different, case-specific tool behaviour without having to reprogram the automation system. This implies rule based model modification (D2.1.1)	MUST
Functional	**(Req-ELW-DLW-5)** Knowledge merging	Automated merging and integration capabilities of separated, partial ontologies into an overall, system-wide valid ontology to ensure global consistency of engineering concepts (FPP)	MUST

Functional	**(Req-ELW-DLW-6)** Knowledge mapping	Automated mapping capabilities of partial ontologies from one representation format into other data formats by means of import and/or export filters (FPP)	**SHOULD**
Functional	**(Req-ELW-DLW-7)** Model generation	Model generation of all necessary disciplinary engineering analysis models by compilation of the design language into consistent, domain-specific model representations. (FPP)	**SHOULD**
Functional	**(Req-ELW-DLW-9)** Traceability	The ELW should be able to create tools with full traceability of the product design – e.g. for certification and future re-use (D2.1.1)	**SHOULD**
Functional	**(Req-ELW-DLW-10)** Design Optimization	Interface and integration of design optimization loops via generic/abstract optimization “adaptors” coupling the design language components to the optimizer capability. (FPP)	**SHOULD**
Functional	**(Req-ELW-DLW-11)** Analysis Tools	Ability to develop automation tools that can evaluate and analyse product models (D2.1.1)	**COULD**
Security	**(Req-ELW-DLW-12)** Secure Access	Considering the sensitive nature of software source code and design data and tools the access to it could be restricted by different user accounts (if there are different user types using the same development system) (D2.1.2)	**SHOULD**
Functional	**(Req-ELW-DLW-13)** Versioning	Versioning should be supported throughout the system including the requirements, source code and the created engineering services to have the flexibility to switch to an alternative design solution (D2.1.2)	**SHOULD**

3.2 High Level Design Languages

3.2.1 Description

Engineering design knowledge needs formalization to be re-useable. This formalization will be designed and developed into appropriate domain specific ontologies and representations using generic and existing ontologies. These ontologies will be captured in Domain Specific Languages using High Level Design Languages (HLDL). HLDLs will cover the knowledge for the various use cases, domains and disciplines and will therefore form the building blocks for the engineering services and workflows in WP3. Representations of physics, structural design and analysis, electrical design and analysis, cost, weight, manufacturing and process knowledge will be the content of these languages.

An abstract geometry ontology allows the mapping of the geometry information to different distinct CAD modellers and should support a vendor neutral CAD geometry representation and is of importance to the different domains and use cases. Implementation of high level design language components do also include methods for a generic automated 3D routing service, an automated finite element analysis and the description of business- and simulation workflows.

This includes:

- Implementation of a dedicated routing graph-based design language for the modelling of use case 3 (3 weeks cockpit) which can interact with other graph-based design languages which express other engineering design tasks.

- Implementation of an interface in an Engineering Service Development Toolkit for a finite element solver for use case 1 (rudder in a month).
- Establishment of a set of test examples which allow for the establishment of automatic testing of individual ontology mapping and routing features.
- Standardized language to express business and simulation workflows

3.2.2 Component requirements

Table 2 is a summary of the high-level requirements derived from the discussions above.

Table 2: Requirements of the High Level Design Languages

Area	(Identifier) Requirement	Description	Classification
Ontology	**(Req-ELW-DSL-1)** Hybrid Workflow Ontology	A set of semantic models and ontologies **must** be delivered to interconnect the distributed knowledge bases for hybrid workflows and related product models in a flexible and scalable manner	**MUST**
Geometry GBDL	**(Req-ELW-DSL-2)** Geometry GBDL basics	Abstract geometry ontology representation in a design language including the demonstration of mappings (i.e. translation capabilities) of abstract geometry elements to distinct domain-specific geometry representations in distinct domain-specific languages.	**MUST**
Geometry GBDL	**(Req-ELW-DSL-3)** Geometry GBDL constraints	Together with an abstract geometry, an abstract way of representing geometrical constraints **should** be developed. It allows the positioning of geometry components in respect to each other.	**SHOULD**
Geometry GBDL	**(Req-ELW-DSL-5)** Geometry GBDL verification	Verification of correct geometry construction by means of dedicated test grammars which system-atically test the defined design language features.	**COULD**
Physics GBDL	**(Req-ELW-DSL-6)** Physics GBDL interface	Implementation of an interface in an Engineering Service Development Toolkit to a finite element solver	**MUST**
Physics GBDL	**(Req-ELW-DSL-7)** Physics GBDL basics	Physics properties **must** be represented and mapped to different target systems. Demonstration and extension capability of an abstract physics ontology representation.	**MUST**
Physics GBDL	**(Req-ELW-DSL-8)** Physics GBDL constraints	Together with an abstract geometry, an abstract way of representing physical boundary conditions **should** be developed. It allows the expression of physical properties related to abstract geometry.	**SHOULD**
Physics GBDL	**(Req-ELW-DSL-9)** Physics GBDL meshing	Propagation of the physical properties and enrichment of an automatically generated mesh with these boundary conditions in an appropriate domain-specific representation suited for engineering analysis and simulation such as finite element (FEM)	**SHOULD**
Routing GBDL	**(Req-ELW-DSL-11)** Routing GBDL basics	A Routing GBDL for representation of cable harnesses **must** be developed which can be coupled with other design languages including related data like electrical schematic information.	**MUST**
Routing GBDL	**(Req-ELW-DSL-12)** Routing GBDL constraints	Together with the Routing GBDL, an abstract way of representing routing related constraints **should** be developed. It allows the use of gradient fields to manipulate the cable harness routing	**SHOULD**

4 Requirements for the Engineering Library

4.1 Engineering Library

4.1.1 Description

The development of an Engineering Library (EL) will take place to rapidly frontload engineering programs based on corporate standards.

The library will be composed of the following main features:

- Engineering services and workflows developed with the ELW (described in 4.2)
- Standard interfaces and exchange formats allowing quick and smooth integration of engineering modules into the appropriate programs (described in 4.3)
- Existing information and predefined solutions (described in 4.4) like
  - design requirements
  - rules and constraints
  - process modules (tasks, deliverables, workflows, human- and simulation-oriented)
  - product modules (parts, assemblies)
- COTS (design) tools

4.1.2 Component requirements

Table 3 is a summary of the high-level requirements derived from the discussions above.

Table 3: Requirements of the Engineering Library

Area	(Identifier) Requirement	Description	Classification
Contents	**(Req-ELW-EL-1)** Engineering Services	The EL **must** contain engineering services made available through KBE and COTS tools for the integration in the AIF (FPP)	MUST
Contents	**(Req-ELW-EL-2)** KBE Tools	The EL **must** contain KBE tools created by the ELW to be used as an engineering service (FPP)	MUST
Contents	**(Req-ELW-EL-3)** Business process workflows	The EL **must** contain business process workflows for the use cases (FPP)	MUST
Contents	**(Req-ELW-EL-4)** Simulation process workflows	The EL **must** contain simulation workflows for the use cases (FPP)	MUST
Contents	**(Req-ELW-EL-5)** Design information	The EL **should** contain (standard) design rules, constraints, materials, design requirements (FPP, D2.1.1)	SHOULD
Contents	**(Req-ELW-EL-6)** COTS Tools	The EL **should** contain wrapped COTS tools for evaluation and analysis of product models to be used as an engineering service (FPP)	SHOULD
Functional	**(Req-ELW-EL-7)** Generic Tools	Automation tools **should** be generic, i.e. non-customer specific (using the same standard solutions) (D2.1.1)	SHOULD
Functional	**(Req-ELW-EL-8)** Visualization Tools	The EL **should** contain tools to provide clear and relevant visualization of the product model (D2.1.1)	SHOULD

Area	(Identifier) Requirement	Description	Classification
Functional	**(Req-ELW-EL-9)** User Interaction	Automation tools **should** allow user interaction if needed (D2.1.1)	SHOULD
Functional	**(Req-ELW-EL-10)** Large Dataset handling	Tools **should** be able to cope with large datasets for evaluation of large amounts of use cases (D2.1.1)	SHOULD
Functional	**(Req-ELW-EL-11)** Standalone Execution	Tools **should** be able to be executed stand-alone (beneficial for debugging, …) (D2.1.1)	SHOULD
Functional	**(Req-ELW-EL-12)** Standardised data exchange	Tools **must** exchange data via standardised interfaces (D2.1.1)	MUST
Functional	**(Req-ELW-EL-13)** Optimization Tool	The EL **must** contain optimisation tools to perform product optimisation, design space exploration, trade studies and to provide clear and relevant visualizations (D2.1.1)	MUST

4.2 Engineering Services

4.2.1 Description

The engineering services in the Engineering Library (EL) are highly use case specific tools, defined to automate one or few specific tasks. These engineering services can have requirements independent from the EL, which are mainly in the functional area. Listing and tracking of the detailed requirement sets related to the individual engineering services is beyond the scope of the document and considered a partner-specific responsibility. Here the engineering services should only be presented with its functions in general.

These engineering services comprise:

- **The Airbus in-house tool Descartes is being extended to allow initialization of a CPACS model** at the beginning of the new conceptual aircraft development process. The intention of this was to provide the means of “sketching” an aircraft model in a 3D CAD environment with as little effort and required input as possible. This way, the conceptual engineer would not have to invest any additional effort into creating such a sketch than he would have to in the traditional approach based on two-side-views. The added value of the approach using Descartes is to allow visualization and evaluation of a 3D CAD model from the very beginning of the design process with a native interface to the parametric CPACS data format. Therefore, a straight forward way of initializing a CPACS model with first data SHOULD be available.
- **An engineering service for automated initialization and synthesis of fighter aircraft configurations.** Using a given set of requirements and a handful of assumptions as input, the service MUST be able to generate a consistent geometry and initial mass breakdown of the aircraft. This is to be achieved through the creation of a knowledge base consisting of a large set of empirical correlations available in design handbooks and from experience. The requirements MUST be automatically read-in from the CPACS data exchange format; resulting information (geometry, masses) MUST be made available to subsequent engineering services using the CPACS format as well. It SHOULD be the case that all major fighter aircraft components and disciplines are covered by the

engineering service and through flexible object-oriented programming based on discipline and component ontologies, it MUST be guaranteed that further disciplines can be added without large effort. Resulting geometry and mass breakdowns MUST be verified to available data of representative fighter aircraft, in case this data can be obtained.

- **Hinge-system Design and Optimization Tool (HDOT)** is a KBE application built using the Python based ParaPy KBE system which enables and automates the hinge system design process. The tool makes use of interfaces to other engineering services in the Engineering Library, specifically the FE solver NASTRAN and Fokker Excel-based stress reserve factor calculation tools. Standard input data and test data used by the engineering service are also provided in the Engineering Library.
  It is part of the Use Case 1B of the Aircraft Design Challenge as described in D2.1.1. It has the ability to carry out exhaustive search of all possible hinge components to determine best hinge assembly satisfying the stress requirements at every hinge location based on cost and/or weight for a given cost and weight model.
  HDOT can quickly and automatically generate a simplified rudder structure based on user defined specifications, generate a mesh based on the rudder structure and carry out structural analysis (using COTS tool) to determine forces acting on the hinges. These forces are in turn essential for the sizing of hinge components at different hinge locations. The main requirements are
    - The rudder outer-mold-line MUST be imported from STEP file.
    - External loads MUST be imported.
    - MUST allow for the generation of different torsion boxes based on 2 or more spar layout.
    - MUST automatically generate torsion box geometry, based on spars and number of ribs using a CAD kernel.
    - MUST automatically mesh the geometry in an external mesher
    - MUST apply loads, extract results and determine critical load cases
    - MUST select standard parts for hinge components from database
    - MUST configure a feasible hinge from standard parts database, that can withstand the loads
    - MUST determine margins of safety, compliance with MS, cost and weight of all parts
- **An engineering service for automated wire harness routing including path smoothing of harness segments.** This is a modularized engineering service comprising automated wire harness pathfinding, routing, and simulation using a multi-body approach in order to get a physical realistic model of a wire harness. Since modifications of the cable and harness placement occur, an additional collision check MUST be included. Physical properties like cable stiffness's SHOULD be taken into account as flexible input parameters within the path smoothing capability. The calculated cable path MUST be iteratively smoothed in real-time according to physical data using a multi-body approach. Physical properties like cable stiffness's SHOULD be taken into account as flexible input parameters.
- **An engineering service for harness stiffness simulation.** A prediction of the mechanical behaviour of cable harnesses for cable routing simulations will be developed. Using the approach of the Finite Element Method (FEM) the harness stiffness for every occurring cross section can be determined. The large variety of cross-sections of cable

harnesses will be categorized. Furthermore, uncertainties regarding geometrical dimensions, material properties or other cable specific information will be investigated. Results will be validated with experimentally measured data (this links to WP5). A concept for the prediction of the mechanical behaviour of cable harnesses for cable routing simulations MUST be developed (using FEM). Calculated values for the prediction of cable harness stiffness MUST be validated with experimentally measured data. The homogenised material data MUST be stored in a data base. A large variety of cross-sections of cable harnesses SHOULD be categorized for investigation of uncertainties regarding geometrical dimensions, material properties, etc. The data SHOULD be usable as input for the multi-body path smoothing approach.

- **An engineering service for the prediction of cable harness segment stiffness's.** A Similarity Prognosis Model using dimensionless parameters is needed to predict wire harness segment stiffness's.
- **An engineering service for the automatic routing of electrical signals within a defined Main Routing Architecture.** This is effectively a Tom-Tom for the EWIS architecture. Based on the Main Routing Architecture, this engineering service routes electrical signals through the main routings based on the most optimal conditions (e.g. length, preferred segregation / separation, etc.). Examples include that system segregations for independence / survivability will need to follow port and starboard sides for system segregations 1 and 2 respectively as a requirement. Note that the number of electrical signals is high and in the order of ten-thousands, by which it is a labor-intensive task when conducted manually. Signals that cannot be routed according all requirements are identified and can be defined manually. The Signal routing module is integrated with the Component selection and Pin assignment engineering services within the detailed electrical engineering design process (explained later).
- **An engineering service to select wire sizes within a network within thermal- and voltage drop constraints.** The Wire size selection engineering service automates the selection of the minimum wire size within voltage drop & thermal constraints. It represents a design problem with a large number of variables and constraints both due to EWIS being a large scale system (high number of signals / wires) as well as the interactions and dependencies within the design to be considered. An example of the latter include interactions between thermal and voltage drop, as the cable resistance is linked to the cable temperature. The Wire Size Selection tool will first include using aerospace standard Thermal rules for wire selection, in the future this will be replaced by advanced thermal design rules as in development within Fokker. Also, integration with the other design tools will enable an even further optimization opportunity.
- **An engineering service for the automatic selection of connectors (including its components) at a production break.** An aircraft can have 100s or in some cases several 1000 connectors. At each production break (an Electrical Wiring Interconnection System (EWIS) is broken down into a multitude of wiring harnesses for reasons of producibility as well as the aircraft being produced in sections as well), connectors are automatically selected to accommodate the wiring at each production break as defined by the Signal routing engineering service. The selection of components includes the connector mating components like contacts, insert arrangements, connector shells, etc. All component are selected within applicable constraints, e.g. environmental characteristics and traded-off based on cost- and weight.

- **An engineering service to assign signals to pins on a connector at a production break.** This engineering service automatically assigns signals (as retrieved by Signal routing) to contacts on connectors at production breaks (as determined by the Component selection engineering service).

4.3 Standard interfaces and exchange formats

4.3.1 Description

In this task data formats and interfaces have to be established which represent the projects knowledge in an integrated manner. The data standards are part of the Engineering Library (see Figure 1).

In aircraft design CPACS (Common Parametric Aircraft Configuration Scheme) is an XML schema definition for efficient data exchange which is currently becoming a quasi-standard across institutions in Europe. Beside product information of multi fidelity-levels, process information is also incorporated within CPACS. This aids in providing settings to the analysis modules with analysis workflows, steering their behaviour according to the project at hand. The following extensions to CPACS are envisioned:

- After identifying the analyses to be performed in light of the aircraft design use cases, CPACS will be extended to cover features required to cover all product information being exchanged between the involved analysis modules.
- Within IDEaliSM, the process information storage capabilities of CPACS will be extended, creating the ability to save process information delivered by the components of the Advanced Integration Framework.
  The possibility of saving data lifecycle information within the central data model will be investigated. In this, the right balance between data size and readability to data reproducibility needs to be found.
- Finally, if needed, automated mapping capabilities for different high level design languages will be developed by establishing in-/export filters in order to link design languages in CPACS.

STEP, defined in ISO 10303, is a widely used set of standards for the description of arbitrary product data that also covers requirements of the aeronautics industry. Most CAD/CAE systems are able to process STEP files. However, their focus is shape data; a lot of the product information is not supported by these systems, which severely limits tools interoperability.

Therefore STEP will be used within IDEaliSM not only with a sub-set of its capabilities, but in a more holistic way.

This includes the:

- Exchange of product model information (CAD, CAE and PLM data) using one or several of the standards STEP ISO 10303-209/233/239/242
- Integration and management of such information from different sources in a consistent database
- Definition of the product structure ontology (STEP ISO 10303)
- Incorporation of KBL and its successor VEC into the list of addressed standards.

STEP is a set of standards that grows as new industry requirements appear. Some of these standards, like AP233 and AP239 apply relatively general data model concepts; these can be specialized by a reference data ontology to meet concrete industrial needs. Else, as STEP is defined by means of the formal data modelling language EXPRESS, standardized as ISO 10303-11, non-standard extensions may be added to STEP data dictionaries to incorporate locally required product information.

4.3.2 Component requirements

Table 4 is a summary of the high-level requirements derived from the discussions above.

Table 4: Requirements of standard interfaces and exchange formats

Area	(Identifier) Requirement	Description	Classification
CPACS	**(Req-ELW-SI-1)** CPACS Use case 1	CPACS **must** cover all product information being exchanged between the involved analysis modules in the aircraft use case	**MUST**
CPACS	**(Req-ELW-SI-2)** CPACS PLM	CPACS **should** include product lifecycle information	**SHOULD**
CPACS	**(Req-ELW-SI-3)** CPACS Process Information	CPACS **should** be extended by process information storage capabilities to save process information delivered by the components of the Advanced Integration Framework	**SHOULD**
KBL	**(Req-ELW-SI-5)** KBL	The wire harness data format KBL **should** be a supported standard	**SHOULD**
VEC	**(Req-ELW-SI-6)** VEC	The holistic wire harness data format VEC **should** be a supported standard	**SHOULD**
STEP	**(Req-ELW-SI-7)** Exchange product model information	The IT-infrastructure **must** have the ability to exchange product model information (CAD, CAE and PLM data) using one or several of the standards STEP ISO 10303-209/233/239/242	**MUST**
STEP	**(Req-ELW-SI-9)** CPACS converter to STEP	The ability **should** be created to convert CPACS to STEP	**SHOULD**
VEC	**(Req-ELW-SI-11)** VEC to KBL converter	The ability **should** be created to convert VEC to KBL	**SHOULD**

4.4 Existing information and predefined solutions

4.4.1 Description

The Engineering Library holds existing information and predefined solutions. To work with this information with automatic engineering services, the data must be well specified. For example the 3D digital mock-up data as geometric boundary conditions is essential for the automatic routing and of wire harnesses. This includes e.g. a meaningful partitioning of the part into assemblies and the addition of relevant electrical data for e.g. the connectors or fixing parts.

4.4.2 Component requirements

Table 5 is a summary of the high-level requirements derived from the discussions above.

Table 5: Requirements to existing information and predefined solutions

Area	(Identifier) Requirement	Description	Classification
Availability	**(Req-ELW-MUD-1)** Availability of realistic geometry	A realistic cockpit geometry from the automotive industry **must** be available, which can be used as geometrical environment (CR2)	**MUST**
Availability	**(Req-ELW-MUD-2)** Availability of electrical data	Electrical data **must** be available to complement the geometrical connectors or fixing parts (CR2)	**MUST**
Availability	**(Req-ELW-MUD-3)** Part partitioning	Meaningful partitioning of parts into assemblies **should** be prepared to separate entities for application of rules (CR2)	**SHOULD**

5 Inventory list of current used data formats

This section is an inventory list of current tools of the solution providers and their supported standards, API's and data formats. Possibilities of interoperability between these tools can be elaborated.

5.1 Fraunhofer LBF

Software application name (version)	HSSC (Harness Segments Stiffness Calculator)
Engineering services provided	Stiffness Calculation of Cable Harness Segments
Operating system (version)	Microsoft Windows 7 Java Runtime Environment 8 ANSYS (R14.5, R15.0, R16.0) Screen resolution > 1200 x 850 pixel
Virtual machine support (version)	Not applicable
Data formats support (version)	Geometry and stiffness information: *.vec
Information model availability, name (version)	Not available
Information modelling language to document the information model	Not available
Programming languages support	Not available
API support	No
Web-services support	No
Provided test data	Tbd in next version
Contact name (email address)	Christoph Tamm (christoph.tamm@lbf.fraunhofer.de)
Other information	

5.2 IILS

Software application name (version)	DesignCompiler43 (version 2.2)
Engineering services provided	Wire Harness Routing Service (3rd version)
Operating system (version)	Windows, Linux (no special version) 64-bit recommended
Virtual machine support (version)	with client operating system Windows or Linux
Data formats support (version)	datasets: *.xls, *.xlsx, *.cvs electrical information: *.kbl, *.vec geometrical information: *.step (AP203, AP214), *.stl, *.vtp
Information model availability, name (version)	own data model(s) based on UML
Information modelling language to document the information model	UML
Programming languages support	Java, (xtend)
API support	No / not yet
Web-services support	No / not yet
Provided test data	none
Contact name (email address)	Marc Eheim (eheim@iils.de)
Other information	Command line execution without GUI possible

5.3 iMinds-DistriNet, KU Leuven

Software application name (version)	Impera
Engineering services provided	Integrated configuration management for automated cloud deployment
Operating system (version)	Linux (CentOS, Fedora, Ubuntu)
Virtual machine support (version)	yes
Data formats support (version)	NA – not applicable
Information model availability, name (version)	NA – not applicable
Information modelling language to document the information model	NA – not applicable
Programming languages support	NA – not applicable
API support	Python
Web-services support	yes
Provided test data	NA
Contact name (email address)	Stefan Walraven (stefan.walraven@cs.kuleuven.be) Bert Lagaisse (bert.lagaisse@kuleuven.be) Bart van Brabant (bart.vanbrabant@cs.kuleuven.be)
Other information	https://github.com/impera-io/impera Support for deploying on OpenStack (private cloud) and Amazon AWS

5.4 Jotne EPM Technology AS

Software application name (version)	EXPRESS Data Manager (EDM)
Engineering services provided	ISO 10303 STEP data exchange, integration and archival
Operating system (version)	Windows/Unix/Linux/MacOs
Virtual machine support (version)	yes
Data formats support (version)	XML(P28), STEP (P21)
Information model availability, name (version)	All ISO 10303-11 application protocols and user defined schemas
Information modelling language to document the information model	EXPRESS
Programming languages support	C/C++, JAVA, .NET, EXPRESS-X
API support	Yes
Web-services support	Yes
Provided test data	GLIDER Aircraft
Contact name (email address)	Kjell Bengtsson (kjell.bengtsson@jotne.com)
Other information	

5.5 KE-works

Software application name (version)	KE-chain v1.3.8
Engineering services provided	Engineering Process Management component in the IDEaliSM Integration Framework
Operating system (version)	Linux based server deployment (Ubuntu-, RHEL-, Debian-based)
Virtual machine support (version)	VMWARE & VirtualBox
Data formats support (version)	Custom
Information model availability, name (version)	Product Information Model, Workflow Information Model
Information modelling language to document the information model	The Workflow Information Model is loosely based on BPMN, the Product Information Model is based on influences from Step & UML object modelling
Programming languages support	Python
API support	-
Web-services support	REST, SOAP
Provided test data	-
Contact name (email address)	Stefan van der Elst (stefan.vanderelst@ke-works.com)
Other information	

5.6 DLR

Software application name (version)	Remote Component Environment (RCE), v6.2.1 and higher
Engineering services provided	A distributed, workflow-driven integration environment in which complex calculation and simulation workflows consisting of existing design and simulation tools on dedicated servers can be created, managed and executed. Libraries to connect analysis modules to the central data model CPACS
Operating system (version)	Red Hat Enterprise Linux 6 Workstation (64 bit) Debian 7 stable (64 bit) SUSE Linux Enterprise Desktop ("SLED") 11 SP2 (64 bit) Windows 7 (64 bit)
Virtual machine support (version)	Possibly, not used up until now
Data formats support (version)	Data formats depend on integrated design and simulation tools Extensions are provided for XML file handling (using xml interfacing (TIXI) and geometry interfacing (TIGL) libraries for CPACS v2.3 and higher)
Information model availability, name (version)	Common Parametric Aircraft Configuration Schema (CPACS), Version 2.3
Information modelling language to document the information model	XSD (XML Schema Definition)
Programming languages support	All languages are supported. Supporting libraries provide interfaces for: C/C++, Python, MATLAB and FORTRAN. Java if required
API support	yes: Java for RCE, C++ for CPACS supporting libraries
Web-services support	
Provided test data	Internally developed medium-range transport aircraft described in CPACS, VAMPzero conceptual design tool + GUI interface embedded in RCE
Contact name (email	Erwin Moerland (erwin.moerland@dlr.de),

address)	Thomas Zill (thomas.zill@dlr.de)
Other information	Contact persons of DLR's software department: Doreen Seider (doreen.seider@dlr.de), Robert Mischke (robert.mischke@dlr.de)

Software application name (version)	Multiple Aircraft Analysis Tools
Engineering services provided	Disciplinary analyses for aircraft conceptual and pre-design purposes Libraries to connect analysis modules to the central data model CPACS
Operating system (version)	Mostly Windows 7 (64 bit), some Linux
Virtual machine support (version)	Possibly, not used up until now
Data formats support (version)	All support CPACS v2.3
Information model availability, name (version)	Common Parametric Aircraft Configuration Schema (CPACS), Version 2.3
Information modelling language to document the information model	XSD (XML Schema Definition)
Programming languages support	All languages are supported. Supporting libraries provide interfaces for: C/C++, Python, MATLAB and FORTRAN. Java if required
API support	C++ for CPACS supporting libraries
Web-services support	
Provided test data	Internally developed medium-range transport aircraft described in CPACS
Contact name (email address)	Erwin Moerland (erwin.moerland@dlr.de), Thomas Zill (thomas.zill@dlr.de)
Other information	Software tools remain the proprietary of the tool developer,

	therefore individual tool contact persons vary throughout DLR

5.7 NOESIS Solutions

Software application name (version)	Noesis Optimus 10.16 and higher Noesis id8 enterprise web platform
Engineering services provided	A commercial off the shelf product integration and design optimization tool for complex and distributed multidisciplinary optimization problems. Provides simulation workflows, design and analysis methods for exploration and optimization, surrogate modelling for model-based predictions, robustness and reliability analysis, uncertainty quantification. Interfaces are provided to most commonly used commercial tools, provides inclusion and extension of optimization and metamodeling features, fully scriptable in Python 2.7. Already established in major aeronautic and automotive industry. Full support to CPACS and any XML structured format available.
Operating system (version)	Windows Server 2003 on x86 and x86-64 (both AMD & Intel hardware) Windows Vista on x86 and x86-64 (both AMD & Intel hardware) Windows Server 2008 on x86 and x86-64 (both AMD & Intel hardware) Windows 7 on x86 and x86-64 (both AMD & Intel hardware) Windows 8/8.1 on x86 and x86-64 (both AMD & Intel hardware) Linux SUSE Enterprise 10.3 and higher on x86 and x86-64 (native 64-bit supported) Linux RedHat Enterprise 5, 6 and 7 on x86 and x86-64 (native 64-bit supported) Linux CentOS 5, 6 and 7 on x86 and x86-64 (native 64-bit supported)
Virtual machine support (version)	Yes, all virtualization engines compatible with the operating systems above + UBUNTU
Data formats support (version)	CATIA, MATLAB, LMS Virtual.Lab, Ricardo Wave, MS Excel, LMS Imagine.Lab, ANSYS Workbench, ANSA, LS-Dyna, Sigmetrix, PTC Pro/E 4 and 5, XML Generic, Moldflow, SpaceClaim, CoCreate, CD-Adapco Star CCM+, Calc (Linux Excel), JMAG, Siemens NX (CAD+CAE), MapleSim, Maple, AVL Excite/Boost, MSC Nastran OP2, Samcef, GT Power, SimulationX, MSC Adams Cars/View, Flowmaster, Abaqus, MSC Nastran bulk (f06, blk)

Information model availability, name (version)	Workflow XML 1.0
Information modelling language to document the information model	Workflow XML (WFXML, based on a specific XSD grammar)
Programming languages support	C++, Python
API support	Python
Web-services support	Can connect to REST services as client. The id8 platform supports remote web service operations (workflows storage, simulation data post-processing, RSM creation and execution). A web interface (Optimus Workflow Manager) has been implemented to support more advanced remote operations (execution of workflow methods, partial workflow materialization) through REST API.
Provided test data	none
Contact name (email address)	Roberto d'Ippolito (roberto.dippolito@noesissolutions.com)
Other information	

6 Standard Interfaces and Data Formats

Exchanging knowledge in a consistent way is fundamental to the success of the integration project. The consortium has identified several data formats in which the central product model could be described. These data formats and standards might be used as interchange formats between the different engineering services and tasks. This section starts with a description of the master data management (MDM) module, performing the central management of data within the AIF.

6.1 Standardisation Strategy

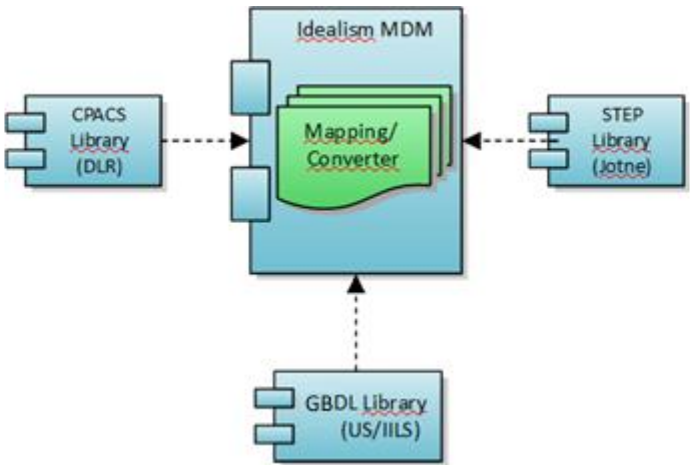


Figure 2: Data formats contributing to the MDM

In order to guarantee consistent data management in WP4, a master data management (MDM) module is established, depicted in Figure 2. This MDM is capable of providing data to the other modules within the AIF, in the data format requested by the implementation. This implies the data types used through the MDM can differ from one use case to the other. If the implementation of a use-case requires exchanging information between the involved data standards, converter tools are established, where necessary, aiding in the translation from the one to the other.

For example, in the case of graph-based design languages, the developed ontologies act as interfaces to standard data formats (like STEP). These ontologies are published and maintained by the graph-based design language developers (using the engineering workbench) and integrated into the standard data formats of the master data model.

6.2 BPMN

The Business Process Model and Notation (BPMN) is a widely accepted standard for modelling business processes but also technical workflows. Initially, BPMN was a standard that only specified how a process can be visualized in a diagram but since its latest version 2.0 it also specifies a formal data representation that allows for a standardized exchange of process models. Since IDEaliSM aims to create an Integration Framework to integrate multiple disciplines, departments, sites and even companies, process models play a major role in the project. Hence, BPMN is highly relevant for the project. BPMN 2.0[4] will be used in the IDEaliSM framework by the Engineering Process Management module and in interaction with the simulation workflow module.

In BPMN a process consists of multiple activities and events (incl. it's starting point and its end) that are structured in a sequential flow (that may also feature parallel and/or alternative process flows). It also allows for modelling organizational responsibilities for activities using the mechanism of swim lanes (a visualization approach that is mainly targeted at a management audience) and one may specify documents and/or development artefacts as inputs and outputs of activities. Nesting of processes is also possible. Finally, BPMN provides a set of specialized modelling elements for specifying details that are only relevant for workflow management (such as email notification events or task timeouts, etc.).

Since BPMN, as a data format, is not only meant for exchanging process models but also for exchanging process diagrams, it also features information about the visualization of process elements as an integral part of its data representation. These elements are irrelevant for the IDEaliSM project. There must be investigated how the BPMN model relates to the information in the other models used in the tool in the IDEaliSM framework.

6.3 OWL

The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies. Ontologies are a formal way to describe taxonomies and classification networks, essentially defining the structure of knowledge for various domains: the nouns representing classes of objects and the verbs representing relations between the objects. Ontologies resemble class hierarchies in object-oriented programming but there are several critical differences. Class hierarchies are meant to represent structures used in source code that evolve fairly slowly (typically monthly revisions) whereas ontologies are meant to represent information on the Internet and are expected to be evolving almost constantly. Similarly, ontologies are typically far more flexible as they are meant to represent information on the Internet coming from all sorts of heterogeneous data sources. Class hierarchies on the other hand are meant to be fairly static and rely on far less diverse and more structured sources of data such as corporate databases.
The OWL languages are characterized by formal semantics. They are built upon a W3C XML standard for objects called the Resource Description Framework (RDF).

6.4 STEP – ISO 10303

The growing need for interoperability of different CAD-systems resulted in the initial release of the ISO 10303 standard in 1994 under its title: "Industrial automation systems and integration - Product data representation and exchange". Today the Standard for the Exchange of Product Model Data (STEP) – as ISO 10303 is often informally referred to - is well tested and widely used daily, especially in the CAD area. STEP, however, covers not only most of the scope of current CAD-systems, but also most of the remaining data needed to describe a product during its

[4] http://www.omg.org/spec/BPMN/2.0/

lifecycle, such as analysis, manufacturing and operational data. Not all of the STEP capabilities are supported by commercial actors today.

Figure 3 illustrates the development of and its coverage of industrial data over the years: The STEP standard

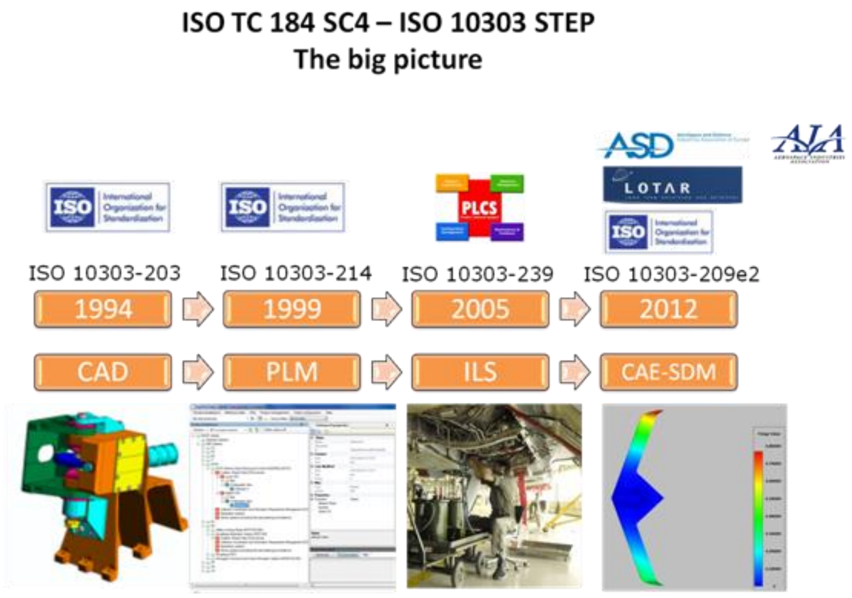


Figure 3: The development of the STEP standard over the years

There are the following reasons for the good uptake of STEP by industry:

- STEP can represent volume models with the required industrial accuracy and quality;
- STEP integrates product shape with other product properties and life-cycle information;
- STEP is a formal data model specified by the language EXPRESS (ISO 10303-11), which is among the most powerful data modelling languages with respect to constraining a model; this enables high data quality due to automated data verification and validation;
- STEP is not only an information model, but defines also several implementation methods, such as, file formats and database access interfaces;
- STEP has a framework for testing of vendor translators, CAx-IF (implementers forum);
- STEP has no serious competitors.

STEP is not a single document, but a series of standards; each document is called a Part. The following Part-numbering system has been imposed on ISO 10303 for its various aspects:

Part 1	: Overview and fundamental principles
Parts 10-19	: Description methods
Parts 20-29	: Implementation methods
Parts 30-39	: Conformance testing methodology and framework
Parts 40-99	: Integrated generic resources
Parts 100-199	: Integrated application resources
Parts 200-299	: Application protocols
Parts 300-399	: Abstract test suites
Parts 400-499	: Application Protocol Modules
Parts 500-999	: Application interpreted constructs
Parts 1000-2999	: Application modules
Parts 3000-...	: Business Object Models.

Additional details of ISO 10303 are included in Annex A: Step on a page.

For IDEaliSM mainly APs 209, 239 and 242 are of interest as they cover the industry domains of the IDEaliSM partners and have considerable commercial support.

7 Commonly used data formats

This chapter contains wide-spread data formats, which are used in design processes every day. However these data formats are not a standard yet.

There are multiple different data formats used within the project for different engineering domains (geometry, FEM, etc.) ranging from openly available formats like IGES to proprietary formats like CatPart for CATIA or input files for Patran and Nastran.

In the following only those data formats are described which are extended during the project or for which converters are created. These common information formats are often based on XML (see Annex B:).

7.1 CPACS

The conceptual and preliminary phases of aircraft design ranging up to high fidelity Multidisciplinary Design Optimization (MDO) are characterized by their interdisciplinary character as well as by an agile way of collaboration between heterogeneous partners. Agility goes in line with the frequent establishment of links between analysis services. In this context the XML schema CPACS (Common Parametric Aircraft Configuration Schema) was developed by DLR to establish these links with minimum effort.

CPACS is a data definition for the air transportation system. Using a central model approach, the number of interfaces between analysis modules within a design system is decreased significantly, as shown in Figure 4. Furthermore, by adhering to a standard for data exchange, exchanging analysis modules within a design process is significantly simplified.

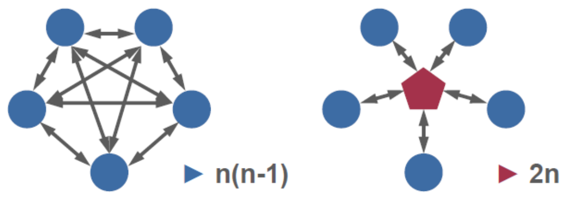


Figure 4: A Central Model Approach significantly reduces the amount of interfaces within a design process

The development of CPACS for aircraft design began in 2005. CPACS enables engineers to exchange information between their tools. It is therefore a driver for multi-disciplinary and multi-fidelity design in distributed environments. CPACS describes the characteristics of aircraft, rotorcraft, engines, climate impact, fleets and mission in a structured, hierarchical manner. Not only product but also process information is stored in CPACS. The process information helps in setting up workflows for analysis modules. The scope is by now enlarged to take into account topics such as high-lift, noise and climate impact, engine design and air transportation system modelling. CPACS can be combined with existing aircraft design systems.

Several analysis modules are connected to CPACS. An example of information extracted by multiple disciplinary analysis modules is shown in the Figure 5. Different models for structure, aerodynamic and load analysis can be derived from the same file. As all models are derived from the same data it is assured that they rely on the same references, i.e. geometry. Multi-disciplinary processes are therefore enhanced from central model applications.

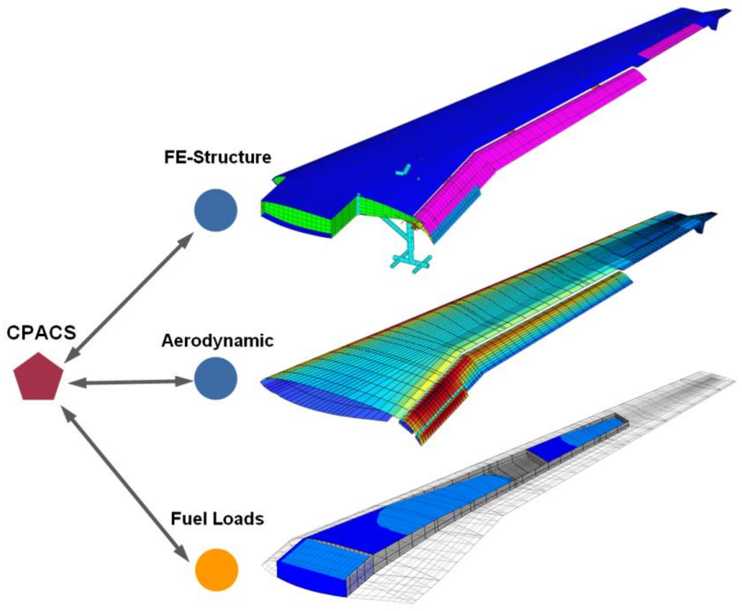


Figure 5: Example of multi-disciplinary analysis using CPACS

Furthermore, CPACS is a hierarchic data structure therefore it is possible to work on different levels of fidelity. The deeper the structure the more detail is present.

As CPACS is a medium for communication it is supposed to be an open standard. It is available as Open Source Software under the Apache 2.0 license and further information can be found at https://software.dlr.de/p/cpacs/home/.

7.2 GBDL

Graph-Based Design Languages are a way of supporting the activity of engineering design. They are inspired by natural human languages, in which the vocabulary (i.e. the words) and the rules (i.e. the building laws) define a so-called language grammar. This means that any correct sentence in this language (i.e. a permissible vocabulary combination) represents a valid engineering product variant.

The increase in productivity, higher model quality and shorter time-to-market stems from modelling and processing the design knowledge on a higher level of abstraction then done previously using *model-to model transformations*. The mapping of this abstract level into a specific data format is provided by *model-to-text transformations*. This avoids an intermixing of the *per se* pure, product specific design knowledge with vendor-specific representation dependencies.

GBDLs on the basis of the internationally standardized *Unified Modeling Language (UML)* possess therefore a well distinct information processing concept and are therefore easily readable, editable and storable based on publicly available UML tools.

These are developed and part of the Engineering Language Workbench and are therefore described in deliverable D4.3.1 [8] in section 3.2.1 in more detail.

7.3 KBL[5]

Innovations in automotive industry like adaptive cruise control or multimedia passenger entertainment systems nowadays define themselves by electric and electronic components. As the electrical wiring system builds the essential infrastructure for automobile electronics, the wire harness becomes increasingly complex. This need for increased complexity comes along with the minimizing of design time and shortening of lead times.

Therefore the collaboration of car manufacturers and harness suppliers is a challenge. The traditional way that a supplier receives harness design data from the car manufacturer has to change. Instead of various drawings and lists in proprietary formats he needs a specification, which describes the wire harness in its entirety so that the manufacturer can plan the manufacturing and build the harness, based on the data he receives. Such a specification should be based on standards to fulfil the requirements for open development partnerships.

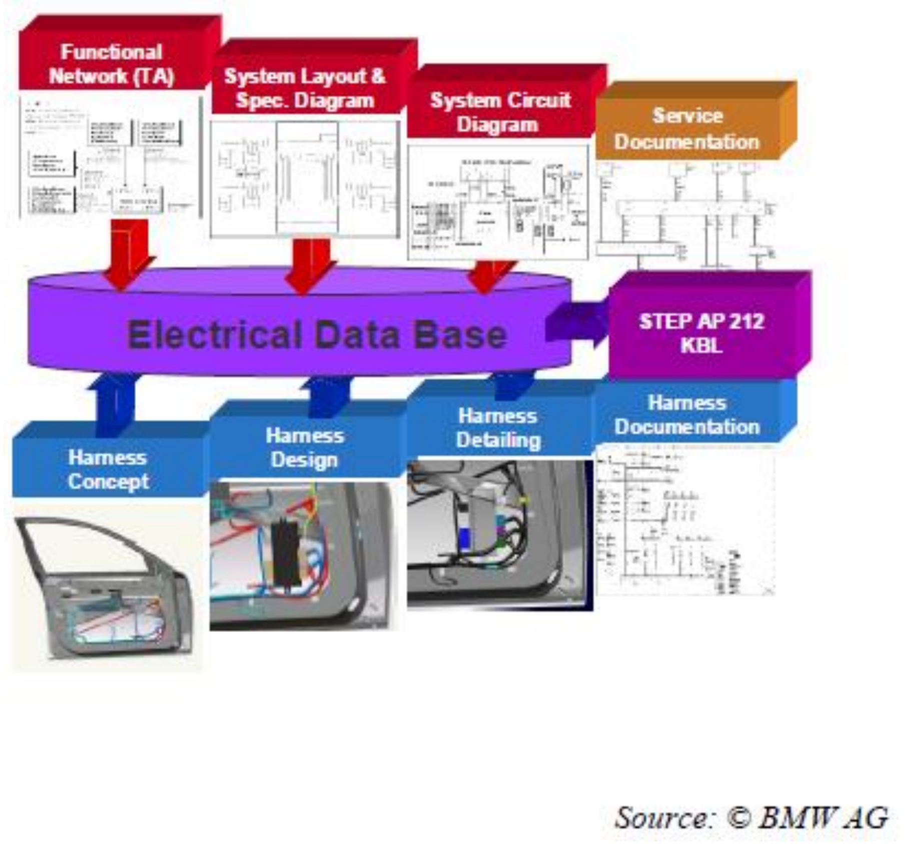


Figure 6: Harness Design Process [10]

[5] http://ecad-wiki.prostep.org/doku.php?id=specifications:kbl

The objectives of the VDA Working Group “Car Electric” are the harmonization of requirements and the development of recommendations for the exchange of product data in the area of car electrical systems.

This recommendation is a result of the working group and has been developed with the participation of major OEMs and harness suppliers: The “Harness Description List” [10]. This specification is also known under the name “KBL”, which stands for “Kabelbaumliste”, the German translation for “Harness Description List”. The recommendation defines how harness design data coming from various sources like 3D CAD systems or CAE system can be represented in an aggregated view.

The newest version of the recommendation, also called KBL 2.4, is a bridge release. Its objective is to enable a smooth migration from KBL to VEC:

- Lower the implementation hurdle for VEC, especially for the supplier interface
- Define the migration path to VEC
- Extensions to KBL 2.4 to enable the interoperation with VEC modeling
- All new KBL concepts are addressed by VEC, too
- Keep KBL scope (physical harness)

The data format is described in more detail in the document ‘Harness Description List (KBL)’ [10], accessible via:

https://www.vda.de/de/services/Publikationen/Publikation.~1267~.html

7.4 VEC[6]

The complexity of today's vehicle electrical systems is constantly growing. A vast variety of options is on the market. Firmly organized and integrated cross-company development processes are essential, combined with powerful, integrated IT infrastructures to support all cross stakeholders.

Against this background, the joint VDA and ProSTEP iViP Association project group "Process Chain Car Electric" has developed standardised data formats for the uniform description of wiring harnesses and related data. Providing the Harness Description List (KBL, VDA 4964 [10]) and supplementing schemas was a leap forward with regard to the improvement of car electric development processes and their integration in the development processes for complete vehicles.

But for supporting the whole car electric development processes the provided specifications were not sufficient. Therefore additional use cases have to be addressed. The objective of the joint VDA and ProSTEP iViP Association project group "Process Chain Car Electric" was against this background to name these use cases and on that basis specify the Vehicle Electric Container (VEC, VDA 4968 [11]) as the required standardised data format in this context.

In the end, the VEC data format specification harmonizes and integrates the already existing solutions with the newly gathered requirements. The VEC data format specification addresses a significantly extended field of application, focussing not only on one single wiring harness but on the whole electric system. The VEC data format specification is capable of supporting a huge amount of data exchange use cases all along the electric system development process.

[6] http://ecad-wiki.prostep.org/doku.php?id=specifications:vec:start

Focus within the VEC specification was to address automotive requirements. But it is also expected that the available VEC specification addresses the needs of the aerospace industry as well.

The data format is described in more detail in the document ‘Vehicle Electric Container (VEC)’ [11], accessible via:

https://www.vda.de/de/services/Publikationen/vehicle-electric-container-vec.html

8 Conclusion

The objective of WP4 is to design and develop an Engineering Language Workbench to enable the creation of automated engineering services and workflows as well as an Engineering Library to hold all the building blocks that can be used in the Advanced Integration Framework.

This document discusses the data standards which are required to build the services to seamlessly integrate the services into the AIF. Furthermore, the requirements for both, the ELW and the EL are specified and added to the document.

Compared to the previous version of the document some minor changes were made due to the feedback loop from the validation and verification of the demonstrators [6]. Seven requirements were removed, that are not applicable by the use cases. In Annex D: these requirements can be accessed for traceability reasons, of which 2 are SHOULD, and 5 are COULD. One requirement is obsolete because it is no longer necessary due to improvement of engineering services (Req-ELW-DSL-4). The remaining 6 are just not needed in the defined use-cases. Furthermore, the requirements for the specific functions of the engineering services were removed, since these are out-of-scope in this document and should the responsibility of the associated partners.

Since no new user stories are defined in the final Use-case Specification [5], no new requirements emerge to cover new user needs. Considering completeness in translating the user stories into requirements, one missing requirement was identified and added (Req-ELW-SI-11).

This document contains the bases for creating the third and final iteration of the engineering capabilities emerging from ELW and EL which will be documented in D4.2.3 and D4.3.3. These capabilities will be finally industrially validated in deliverable D5.1.3.

9 References

[1] IDEaliSM, „Full Project Proposal,“ 2014.

[2] IDEaliSM, „Change Request (CR) #2: Integrated & Distributed Engineering Services framework for MDO,“ 2016.

[3] IDEaliSM, „D2.1.1 Use-case Specification - Baseline,“ 2015.

[4] IDEaliSM, „D2.1.2 Use-case Specification - Update,“ 2016.

[5] IDEaliSM, „D2.1.3 Use-case Specification - Final,“ 2017.

[6] IDEaliSM, „D5.1.2 Integration framework validation - Update,“ 2017.

[7] B. Arthur, The Nature of Technology - What It Is and How It Evolves, New York: Free Press, 2009.

[8] IDEaliSM, „D4.3.1 Engineering Language Workbench - Baseline,“ 2016.

[9] IDEaliSM, „D4.3.2 Engineering Language Workbench - Update,“ 2017.

[10] V. d. Automobilindustrie, „Harness Description List (KBL) (VDA-Recommendation 4964 V2),“ 2014.

[11] V. d. Automobilindustrie, „Vehicle Electric Container (VEC) (VDA-Recommendation 4968),“ 2014.

[12] IDEaliSM, „D3.1.2 Requirement Specification for Advanced Integration Framework - Update,“ 2016.

[13] IDEaliSM, „D5.1.1 Integration framework validation - Baseline,“ 2016.

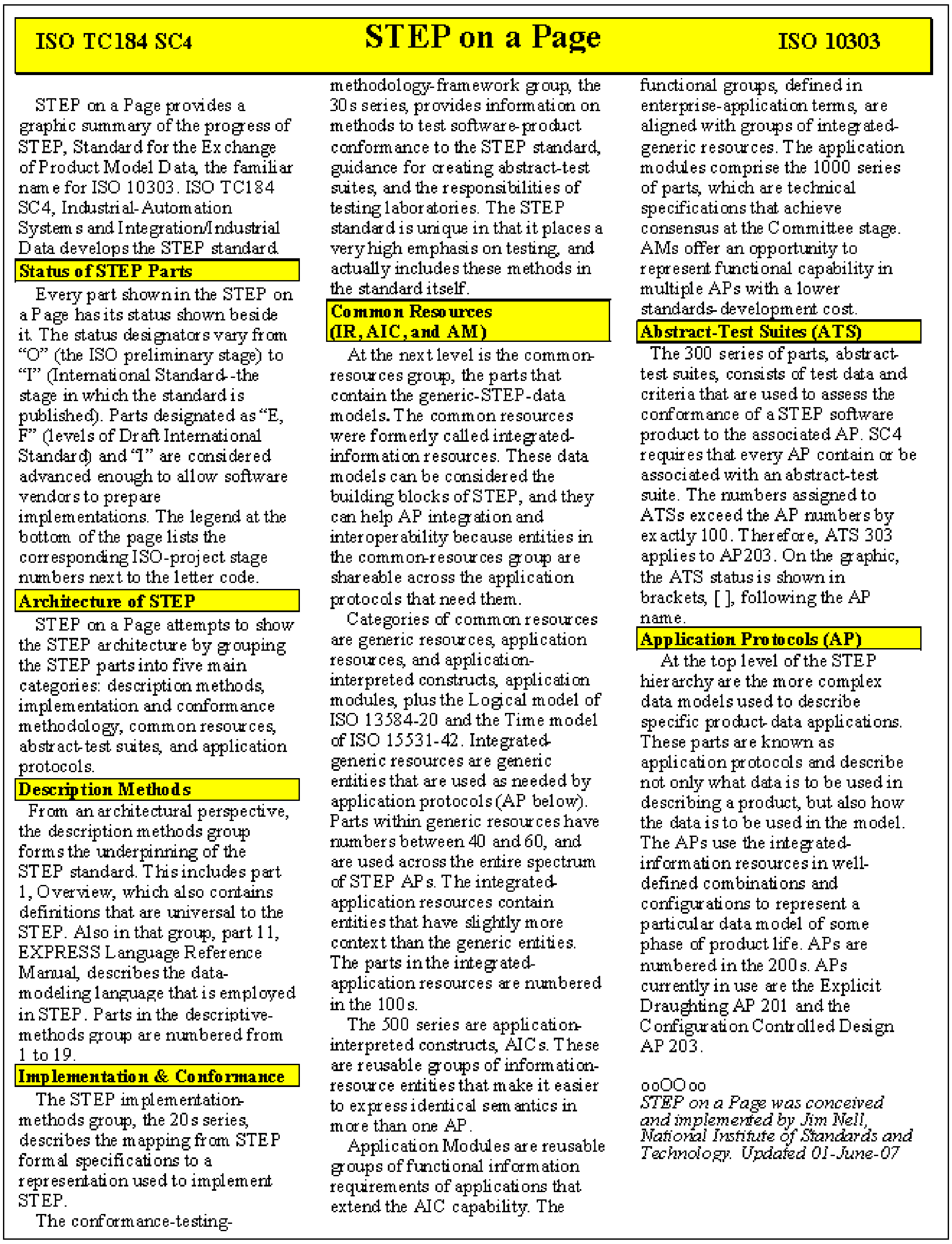
[14] IDEaliSM, „D4.2.1 Engineering Library - Baseline,“ 2016.

[15] IDEaliSM, „D4.2.2 Engineering Library - Update,“ 2017.

[16] IDEaliSM, „D3.1.3 Requirement Specification for Advanced Integration Framework - Final,“ 2017.

Annex A: Step on a page

This annex includes a summary of ISO 10303 STEP on three pages: a description, an overview of resource parts and an overview of modules. These documents are maintained by NIST, USA (http://www.mel.nist.gov/sc5/soap/).



ISO TC184 SC4 **STEP on a Page** **ISO 10303**

STEP on a Page provides a graphic summary of the progress of STEP, Standard for the Exchange of Product Model Data, the familiar name for ISO 10303. ISO TC184 SC4, Industrial-Automation Systems and Integration/Industrial Data develops the STEP standard.

Status of STEP Parts

Every part shown in the STEP on a Page has its status shown beside it. The status designators vary from "O" (the ISO preliminary stage) to "I" (International Standard--the stage in which the standard is published). Parts designated as "E, F" (levels of Draft International Standard) and "I" are considered advanced enough to allow software vendors to prepare implementations. The legend at the bottom of the page lists the corresponding ISO-project stage numbers next to the letter code.

Architecture of STEP

STEP on a Page attempts to show the STEP architecture by grouping the STEP parts into five main categories: description methods, implementation and conformance methodology, common resources, abstract-test suites, and application protocols.

Description Methods

From an architectural perspective, the description methods group forms the underpinning of the STEP standard. This includes part 1, Overview, which also contains definitions that are universal to the STEP. Also in that group, part 11, EXPRESS Language Reference Manual, describes the data-modeling language that is employed in STEP. Parts in the descriptive-methods group are numbered from 1 to 19.

Implementation & Conformance

The STEP implementation-methods group, the 20s series, describes the mapping from STEP formal specifications to a representation used to implement STEP.

The conformance-testing-methodology-framework group, the 30s series, provides information on methods to test software-product conformance to the STEP standard, guidance for creating abstract-test suites, and the responsibilities of testing laboratories. The STEP standard is unique in that it places a very high emphasis on testing, and actually includes these methods in the standard itself.

Common Resources (IR, AIC, and AM)

At the next level is the common-resources group, the parts that contain the generic-STEP-data models. The common resources were formerly called integrated-information resources. These data models can be considered the building blocks of STEP, and they can help AP integration and interoperability because entities in the common-resources group are shareable across the application protocols that need them.

Categories of common resources are generic resources, application resources, and application-interpreted constructs, application modules, plus the Logical model of ISO 13584-20 and the Time model of ISO 15531-42. Integrated-generic resources are generic entities that are used as needed by application protocols (AP below). Parts within generic resources have numbers between 40 and 60, and are used across the entire spectrum of STEP APs. The integrated-application resources contain entities that have slightly more context than the generic entities. The parts in the integrated-application resources are numbered in the 100s.

The 500 series are application-interpreted constructs, AICs. These are reusable groups of information-resource entities that make it easier to express identical semantics in more than one AP.

Application Modules are reusable groups of functional information requirements of applications that extend the AIC capability. The functional groups, defined in enterprise-application terms, are aligned with groups of integrated-generic resources. The application modules comprise the 1000 series of parts, which are technical specifications that achieve consensus at the Committee stage. AMs offer an opportunity to represent functional capability in multiple APs with a lower standards-development cost.

Abstract-Test Suites (ATS)

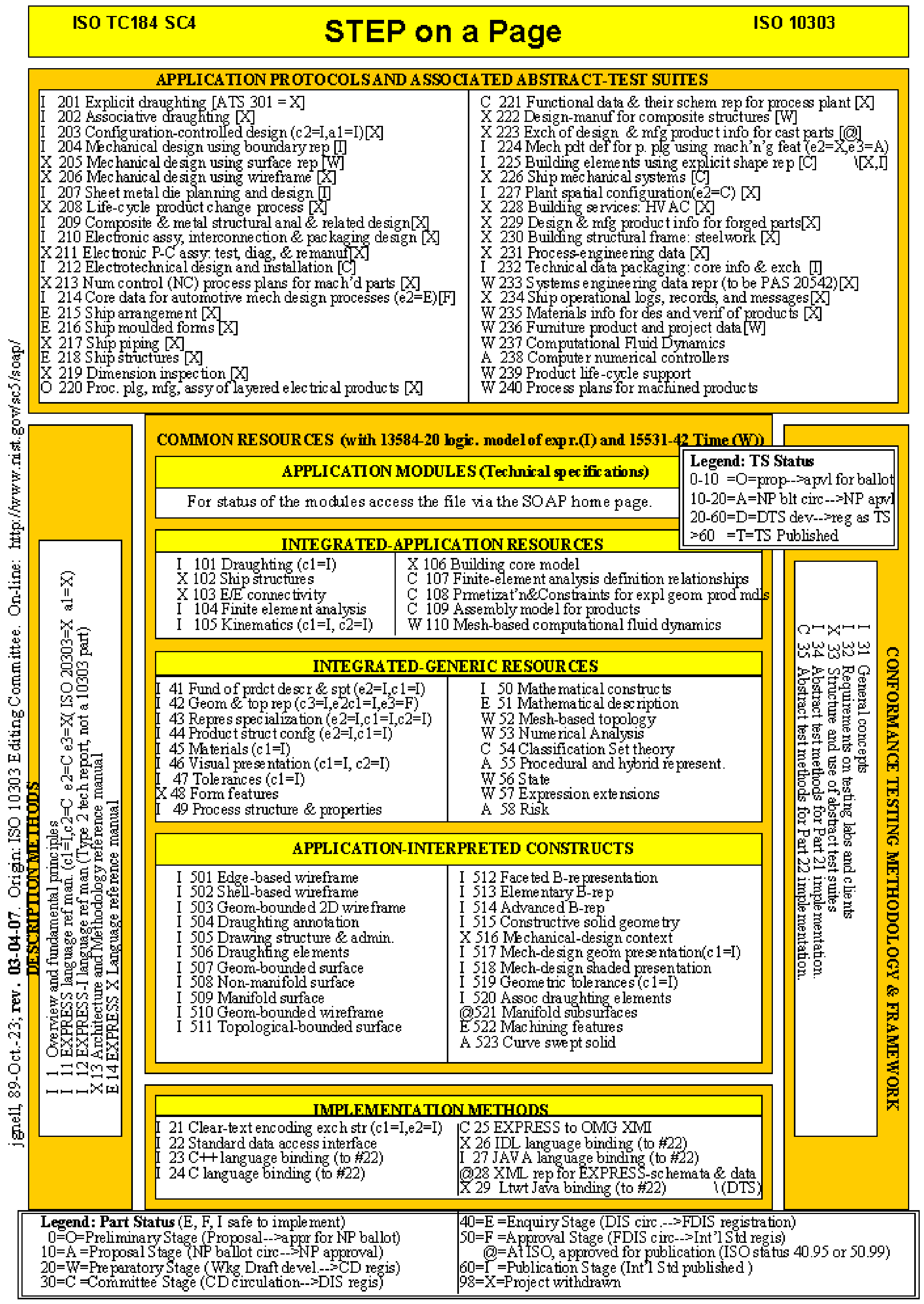
The 300 series of parts, abstract-test suites, consists of test data and criteria that are used to assess the conformance of a STEP software product to the associated AP. SC4 requires that every AP contain or be associated with an abstract-test suite. The numbers assigned to ATSs exceed the AP numbers by exactly 100. Therefore, ATS 303 applies to AP203. On the graphic, the ATS status is shown in brackets, [ ], following the AP name.

Application Protocols (AP)

At the top level of the STEP hierarchy are the more complex data models used to describe specific product-data applications. These parts are known as application protocols and describe not only what data is to be used in describing a product, but also how the data is to be used in the model. The APs use the integrated-information resources in well-defined combinations and configurations to represent a particular data model of some phase of product life. APs are numbered in the 200s. APs currently in use are the Explicit Draughting AP 201 and the Configuration Controlled Design AP 203.

ooOOoo

STEP on a Page was conceived and implemented by Jim Nell, National Institute of Standards and Technology. Updated 01-June-07



ISO TC184 SC4 **STEP on a Page** ISO 10303

APPLICATION PROTOCOLS AND ASSOCIATED ABSTRACT-TEST SUITES

I 201 Explicit draughting [ATS 301 = X]
I 202 Associative draughting [X]
I 203 Configuration-controlled design (c2=I,a1=I)[X]
I 204 Mechanical design using boundary rep [I]
X 205 Mechanical design using surface rep [W]
X 206 Mechanical design using wireframe [X]
I 207 Sheet metal die planning and design [I]
X 208 Life-cycle product change process [X]
I 209 Composite & metal structural anal & related design[X]
I 210 Electronic assy, interconnection & packaging design [X]
X 211 Electronic P-C assy: test, diag, & remanuf[X]
I 212 Electrotechnical design and installation [C]
X 213 Num control (NC) process plans for mach'd parts [X]
I 214 Core data for automotive mech design processes (e2=E)[F]
E 215 Ship arrangement [X]
E 216 Ship moulded forms [X]
X 217 Ship piping [X]
E 218 Ship structures [X]
X 219 Dimension inspection [X]
O 220 Proc. plg, mfg, assy of layered electrical products [X]

C 221 Functional data & their schem rep for process plant [X]
X 222 Design-manuf for composite structures [W]
X 223 Exch of design & mfg product info for cast parts [@]
I 224 Mech pdt def for p. plg using mach'n'g feat (e2=X,e3=A)
I 225 Building elements using explicit shape rep [C] \[X,I]
X 226 Ship mechanical systems [C]
I 227 Plant spatial configuration(e2=C) [X]
X 228 Building services: HVAC [X]
X 229 Design & mfg product info for forged parts[X]
X 230 Building structural frame: steelwork [X]
X 231 Process-engineering data [X]
I 232 Technical data packaging: core info & exch [I]
W 233 Systems engineering data repr (to be PAS 20542)[X]
X 234 Ship operational logs, records, and messages[X]
W 235 Materials info for des and verif of products [X]
W 236 Furniture product and project data [W]
W 237 Computational Fluid Dynamics
A 238 Computer numerical controllers
W 239 Product life-cycle support
W 240 Process plans for machined products

jgnell, 89-Oct.-23; **rev. 03-04-07**. Origin: ISO 10303 Editing Committee. On-line: http://www.nist.gov/sc5/soap/

DESCRIPTION METHODS

I 1 Overview and fundamental principles
I 11 EXPRESS language ref man. (c1=I,c2=C e2=C e3=X(ISO 20303=X a1=X)
I 12 EXPRESS-I language ref man (Type 2 tech report, not a 10303 part)
X 13 Architecture and Methodology reference manual
E 14 EXPRESS X Language reference manual

COMMON RESOURCES (with 13584-20 logic. model of expr.(I) and 15531-42 Time (W))

APPLICATION MODULES (Technical specifications)

For status of the modules access the file via the SOAP home page.

Legend: TS Status
0-10 =O=prop-->apvl for ballot
10-20=A=NP blt circ-->NP apvl
20-60=D=DTS dev-->reg as TS
>60 =T=TS Published

INTEGRATED-APPLICATION RESOURCES

I 101 Draughting (c1=I)
X 102 Ship structures
X 103 E/E connectivity
I 104 Finite element analysis
I 105 Kinematics (c1=I, c2=I)

X 106 Building core model
C 107 Finite-element analysis definition relationships
C 108 Prmetizat'n&Constraints for expl geom prod mdls
C 109 Assembly model for products
W 110 Mesh-based computational fluid dynamics

INTEGRATED-GENERIC RESOURCES

I 41 Fund of prdct descr & spt (e2=I,c1=I)
I 42 Geom & top rep (c3=I,e2c1=I,e3=F)
I 43 Repres specialization (e2=I,c1=I,c2=I)
I 44 Product struct confg (e2=I,c1=I)
I 45 Materials (c1=I)
I 46 Visual presentation (c1=I, c2=I)
I 47 Tolerances (c1=I)
X 48 Form features
I 49 Process structure & properties

I 50 Mathematical constructs
E 51 Mathematical description
W 52 Mesh-based topology
W 53 Numerical Analysis
C 54 Classification Set theory
A 55 Procedural and hybrid represent.
W 56 State
W 57 Expression extensions
A 58 Risk

APPLICATION-INTERPRETED CONSTRUCTS

I 501 Edge-based wireframe
I 502 Shell-based wireframe
I 503 Geom-bounded 2D wireframe
I 504 Draughting annotation
I 505 Drawing structure & admin.
I 506 Draughting elements
I 507 Geom-bounded surface
I 508 Non-manifold surface
I 509 Manifold surface
I 510 Geom-bounded wireframe
I 511 Topological-bounded surface

I 512 Faceted B-representation
I 513 Elementary B-rep
I 514 Advanced B-rep
I 515 Constructive solid geometry
X 516 Mechanical-design context
I 517 Mech-design geom presentation(c1=I)
I 518 Mech-design shaded presentation
I 519 Geometric tolerances (c1=I)
I 520 Assoc draughting elements
@521 Manifold subsurfaces
E 522 Machining features
A 523 Curve swept solid

IMPLEMENTATION METHODS

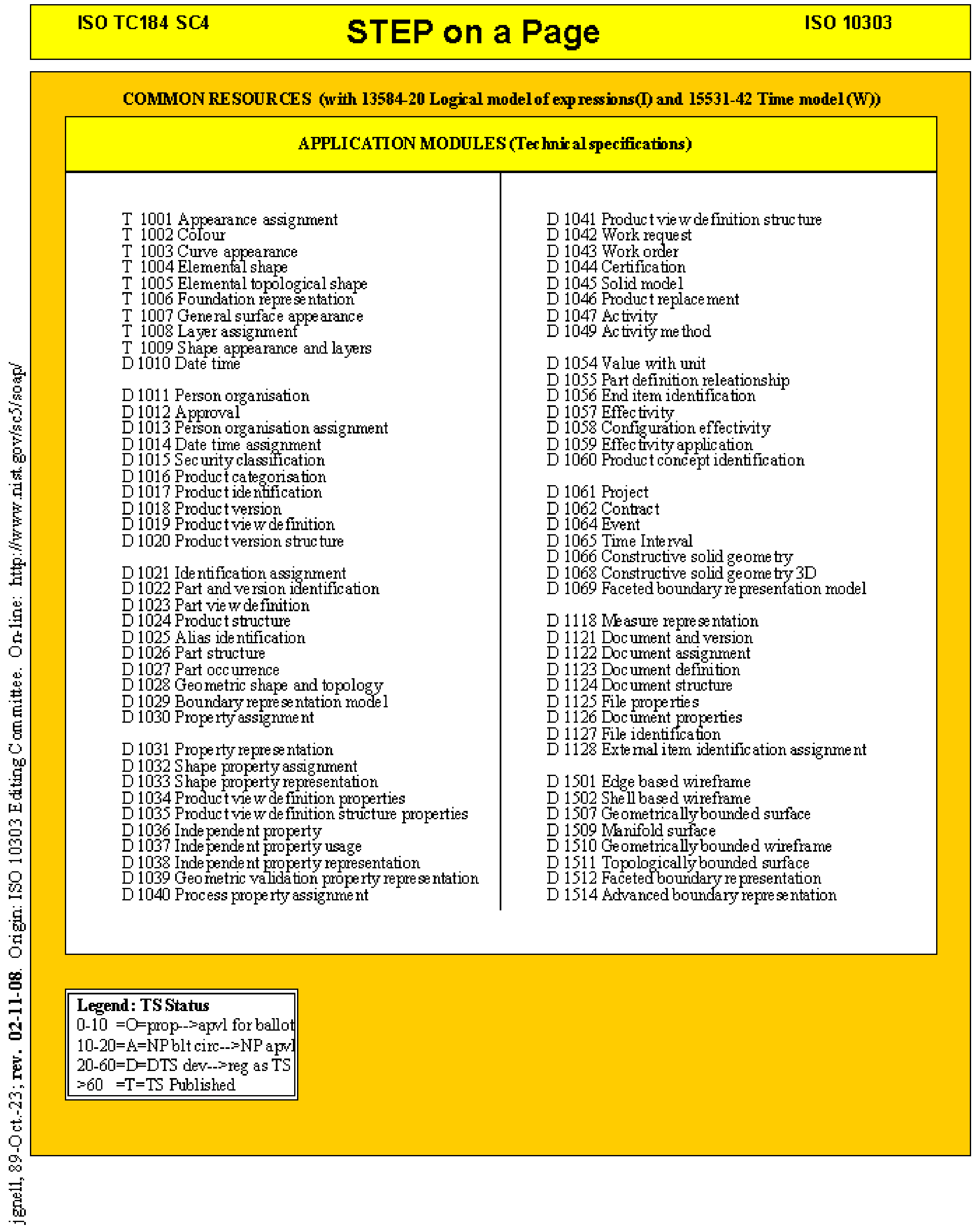
I 21 Clear-text encoding exch str (c1=I,e2=I)
I 22 Standard data access interface
I 23 C++ language binding (to #22)
I 24 C language binding (to #22)

C 25 EXPRESS to OMG XMI
X 26 IDL language binding (to #22)
I 27 JAVA language binding (to #22)
@28 XML rep for EXPRESS-schemata & data
X 29 Ltwt Java binding (to #22) \(DTS)

CONFORMANCE TESTING METHODOLOGY & FRAMEWORK

I 31 General concepts
I 32 Requirements on testing labs and clients
X 33 Structure and use of abstract test suites
I 34 Abstract test methods for Part 21 implementation.
C 35 Abstract test methods for Part 22 implementation.

Legend: Part Status (E, F, I safe to implement)
0=O=Preliminary Stage (Proposal-->appr for NP ballot)
10=A =Proposal Stage (NP ballot circ-->NP approval)
20=W=Preparatory Stage (Wkg Draft devel.-->CD regis)
30=C =Committee Stage (CD circulation-->DIS regis)
40=E =Enquiry Stage (DIS circ.-->FDIS registration)
50=F =Approval Stage (FDIS circ-->Int'l Std regis)
@=At ISO, approved for publication (ISO status 40.95 or 50.99)
60=I =Publication Stage (Int'l Std published)
98=X=Project withdrawn



ISO TC184 SC4 **STEP on a Page** ISO 10303

COMMON RESOURCES (with 13584-20 Logical model of expressions(I) and 15531-42 Time model (W))

APPLICATION MODULES (Technical specifications)

T 1001 Appearance assignment
T 1002 Colour
T 1003 Curve appearance
T 1004 Elemental shape
T 1005 Elemental topological shape
T 1006 Foundation representation
T 1007 General surface appearance
T 1008 Layer assignment
T 1009 Shape appearance and layers
D 1010 Date time

D 1011 Person organisation
D 1012 Approval
D 1013 Person organisation assignment
D 1014 Date time assignment
D 1015 Security classification
D 1016 Product categorisation
D 1017 Product identification
D 1018 Product version
D 1019 Product view definition
D 1020 Product version structure

D 1021 Identification assignment
D 1022 Part and version identification
D 1023 Part view definition
D 1024 Product structure
D 1025 Alias identification
D 1026 Part structure
D 1027 Part occurrence
D 1028 Geometric shape and topology
D 1029 Boundary representation model
D 1030 Property assignment

D 1031 Property representation
D 1032 Shape property assignment
D 1033 Shape property representation
D 1034 Product view definition properties
D 1035 Product view definition structure properties
D 1036 Independent property
D 1037 Independent property usage
D 1038 Independent property representation
D 1039 Geometric validation property representation
D 1040 Process property assignment

D 1041 Product view definition structure
D 1042 Work request
D 1043 Work order
D 1044 Certification
D 1045 Solid model
D 1046 Product replacement
D 1047 Activity
D 1049 Activity method

D 1054 Value with unit
D 1055 Part definition releationship
D 1056 End item identification
D 1057 Effectivity
D 1058 Configuration effectivity
D 1059 Effectivity application
D 1060 Product concept identification

D 1061 Project
D 1062 Contract
D 1064 Event
D 1065 Time Interval
D 1066 Constructive solid geometry
D 1068 Constructive solid geometry 3D
D 1069 Faceted boundary representation model

D 1118 Measure representation
D 1121 Document and version
D 1122 Document assignment
D 1123 Document definition
D 1124 Document structure
D 1125 File properties
D 1126 Document properties
D 1127 File identification
D 1128 External item identification assignment

D 1501 Edge based wireframe
D 1502 Shell based wireframe
D 1507 Geometrically bounded surface
D 1509 Manifold surface
D 1510 Geometrically bounded wireframe
D 1511 Topologically bounded surface
D 1512 Faceted boundary representation
D 1514 Advanced boundary representation

Legend: TS Status
0-10 =O=prop-->apvl for ballot
10-20=A=NP blt circ-->NP apvl
20-60=D=DTS dev-->reg as TS
>60 =T=TS Published

jgnell, 89-Oct-23; rev. **02-11-08**. Origin: ISO 10303 Editing Committee. On-line: http://www.nist.gov/sc5/soap/

Annex B: XML

The Extensible Markup Language (XML) is a markup language to create common information formats and electronically share structured data using standard ASCII text. XML formats are characterized by their flexibility and simplicity and therefore they are human (and machine) readable. XML is playing an increasingly important role in the exchange of data. For example UML, CPACS and the harness information standards KBL and VEC are formatted using XML.

To formally describe the elements in a XML document, a XML Schema Definition (XSD) can be used. XML Schemas express shared vocabularies and rules for defining the structure, content and semantics of XML documents. To transform the structure of an XML document into an XML document with a different structure, XSLT (Extensible Stylesheet Language Transformations) are usually used. For all three standards (XML, XSD, XSLT), recommendations on usage are provided by the W3C.

Annex C: Requirements classification

Letter	Meaning	Description
M	MUST	Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.
S	SHOULD	Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary.
C	COULD	Describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit.
W	WON'T	Represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future. (Note: occasionally the word "Would" is substituted for "Won't" to give a clearer understanding of this choice).

Annex D: Not applicable requirements

This table shows the requirements that are not deemed applicable to the use-cases and are therefore removed.

Area	(Identifier) Requirement	Description	Classification
Functional	**(Req-ELW-DLW-8)** Round-trip engineering	Round-trip engineering capabilities by means of establishing a potentially permanent and interactive mapping between a domain-specific language and the generic knowledge representation in the design language and/or ontology (FPP)	**COULD**
Geometry GBDL	**(Req-ELW-DSL-4)** Geometry GBDL validation	Validation of correct geometry constructions by checking the water-proof property of the geometry in an automated meshing tool.	**SHOULD**
Physics GBDL	**(Req-ELW-DSL-10)** Physics GBDL mesh validation	Validation of mesh enrichment with physical properties in a FEM-analysis (and optionally a CFD-analysis process) by analysing and comparing the generated simulation results with known reference cases from industry within the provided use cases.	**COULD**
CPACS	**(Req-ELW-SI-4)** Mapping capabilities of GBDL and CPACS	Automated mapping capabilities for different design languages **could** be developed by establishing in-/export filters in order to link GBDL and CPACS	**COULD**
STEP	**(Req-ELW-SI-8)** Consistent database	STEP **should** be used to integrate and manage such information from different sources in a consistent database (single source of truth) (D2.1.1)	**SHOULD**
STEP	**(Req-ELW-SI-10)** Harness information converter to STEP	Converters **could** be created for mapping wire harness information of KBL/VEC to STEP	**COULD**
Availability	**(Req-ELW-MUD-4)** Variants	Different variants for the use case **could** be prepared (CR2)	**COULD**