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: Standard interfaces and exchange formats – baseline

**Author, company:**
Marc Eheim, IILS
Jürgen Freund, University of Stuttgart
Roland Weil, IILS
Stephan Rudolph, University of Stuttgart
Kjell Bentsson, Jotne
Maarten Nelissen, KE-works
Erwin Moerland, DLR
Roberto d'Ippolito, NOESIS
Martin Motzer, DRÄXLMAIER
Kevin van Hoogdalem, KE-works
Jochen Haenisch, Jotne

**Version:**
1.02

**Date:**
August 14, 2015

**Status:**
Final / Released

**Confidentiality:**
Public

## CHANGE LOG

| Vers. | Date | Author | Description |
|---|---|---|---|
| 0.1 | 16.06.2015 | Marc Eheim | Initial Document |
| 0.2 | 18.06.2015 | Marc Eheim | Added thoughts of Freund (U of Stuttgart) |
| 0.3 | 19.06.2015 | Marc Eheim | Added chapter 3 contributed by Weil (IILS) |
| 0.4 | 22.06.2015 | Stephan Rudolph | Major rework of Sections 2.1 and 2.2 |
| 0.5 | 30.06.2015 | Kjell Bentsson | Added section 3.4 "STEP" |
| 0.6 | 09.07.2015 | Marc Eheim | Moved chapter 3 to 4; added new chapter 3: Inventory list of used standards |
| 0.7 | 10.07.2015 | Marc Eheim | Added conclusions chapter |
| 0.8 | 10.07.2015 | Maarten Nelissen | Added section about BPMN |
| 0.9 | 15.07.2015 | Erwin Moerland | Major review of chapters 1 and 2, added information on task 4.3, added DLR contribution to inventory list of used standards in chapter 3, added information concerning section 4.4 "CPACS"<br>(*) extensions were based on version 0.5, used comparison tool of MS word to insert changes made revisions 0.6-0.8. Some revisions are therefore marked inserted by Moerland, Erwin but are made by Marc Eheim and Maarten Nelissen |
| 0.10 | 17.07.2015 | Roberto d'Ippolito | Added section about OWL |
| 0.11 | 21.07.2015 | Roland Weil | Revised amendments/comments by Erwin, cleaned up document, document ready for feview |
| 0.12 | 23.07.2015 | Roland Weil | Minor changes/fixes |
| 0.13 | 24.07.2015 | Martin Motzer | Review |
| 0.14 | 27.07.2015 | Kevin van Hoogdalem | Review |
| 0.15 | 28.07.2015 | Roland Weil | Solved major issues of reviews |
| 0.16 | 30.07.2015 | Jochen Haenisch | Revised all sections related to STEP and Jotne. |
| 0.17 | 31.07.2015 | Roland Weil | Solved minor issues of reviews (2. iteration) |
| 0.18 | 31.07.2015 | Roberto d'Ippolito | Review |
| 1.00 | 31.07.2015 | Roland Weil | Cleaned up document, set status to final |
| 1.01 | 13.08.2015 | Marc Eheim | Moved XML section to annex<br>Moved CPACS and graph-based design languages sections to new chapter "commonly used data formats" |
| 1.02 | 14.08.2015 | Marc Eheim | Change confidentiality to public (approved by all contributors) |

## Table of Contents

# List of Abbreviations

| | |
|---|---|
| **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 |
| **DSL** | Domain specific language |
| **FEM** | Finite element method |
| **IGES** | Initial Graphics Exchange Specification |
| **JT** | Jupiter Tessellation |
| **KBL** | Kabelbaumliste |
| **MDO** | Multi-disciplinary design optimization |
| **OEM** | Original equipment manufacturer |
| **OWL** | Web Ontology Language |
| **PDM** | Product data management |
| **PLM** | Product life-cycle 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 - Flaechenschnittstelle. |
| **VEC** | Vehicle Electrical 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, allowing the automation of complex engineering design tasks. It enables the generation and integration of engineering services and workflows into the Advanced Integration Framework, developed in WP3 of the IDEaliSM project.

The position of the Engineering Language Workbench as well as its relation to the overall design process within IDEaliSM is depicted in Figure 1 The figure also indicates the relation of the framework to the different work packages. The overall setup consists of three main phases:

1. The *Building Phase,* in which the language workbench is used to build a (domain specific) set of tools and information models. These can be used to execute a certain project.
2. The *Configuration Phase*, where the set of tools and models is assembled into fully configured and ready to execute project templates.
3. The *Execution Phase*, where the fully configured project templates are instantiated and executed by the end user. This eventually results in the envisioned final product.

The Engineering Language Workbench is used in the Building Phase to create a set of tools, simulation processes and domain models for a specific engineering task. This task is defined in a template which is fully configured within the Advanced Integration Framework (WP3). In this Configuration phase the configured templates in the workflows of the project are created and are ready for execution. Finally, in the Execution Phase (represented by WP2/WP5 of the IDEaliSM project) the configured project templates are instantiated and executed by the end user to obtain the required project result.

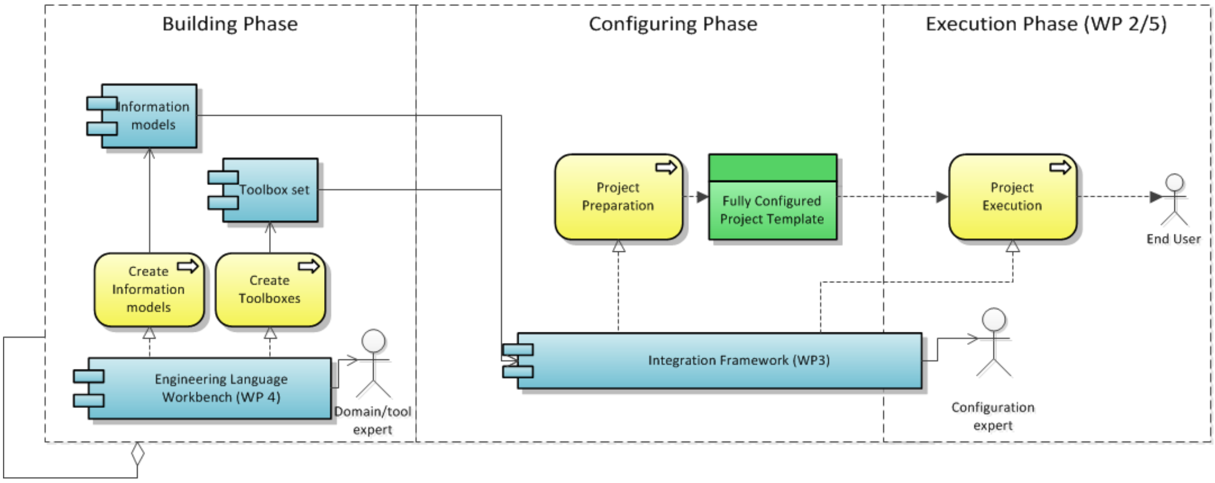


**Figure 1: Overview of the different phases required for the successful execution of a typical engineering project as envisioned within IDEaliSM and the role of the different work packages.**

The Engineering Language Workbench is based on the development of novel so-called “graph-based design languages” together with their background ontologies. To enable the interoperability of the various teams and applications a common knowledge base is needed as well as standardized interfaces and data formats such as CPACS, STEP and UML. Therefore the Engineering Language Workbench contains five sub-components:

1. The Design Language Workbench itself (section 3.1)
2. A set of domain specific and high-level modelling languages (section 3.2)
3. Engineering Library (section 3.3)
4. Standard Interfaces and Exchange Formats (section 3.4)
5. A method for modelling of cable harnesses (section 3.5)

WP4 follows the overall iterative approach in IDEaliSM and thereby delivers three versions of the Engineering Language Workbench 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.

The purpose of this deliverable is to specify the functional and technical requirements for each of the sub-components of the workbench mentioned above. The requirements analysis will be based on the definition of the use cases (WP2). After the industrial validation of the first prototype (WP5), the feedback will be processed and incorporated as new or improved requirements. To conform to the pace of the framework demonstrator D3.2, the requirements will be finalized in three iterations. This to ensure each demonstrator prototype is based on the latest requirements.

This document is organised as follows:

- Section 2 contains a description of the state-of-the-art of data exchange standards (subsection 2.1) and the vision of the standardisation strategy within IDEaliSM (subsection 2.2).
- In Section 3, the different tasks of work package 4 are outlined.
- Section 4 provides an overview of the current state of used tools in the consortium and their supported data formats and standards.
- Section 5 contains a description of the Standard Interfaces and Data Formats which play a key role in the project setup.
- Section 6 summarizes the most important aspects

# 2 Engineering Language Workbench

The Engineering Language Workbench serves the ultimate goal of modelling multi-disciplinary simulation and analysis models and tasks. It therefore heavily relies on a set of domain specific languages (DSLs) and high-level modelling languages as well as ontologies and data standards. These enable a flexible configuration of engineering workflows and services and a straightforward integration into the distributed advanced integration framework. Sections 2.1 and 2.2 describe the current state-of-the-art concerning interfaces and exchange formats and the IDEaliSM vision for achieving the intended project goals.

## 2.1 State-of-the-Art

The design of complex cyber-physical 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 multi-disciplinary 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 involves usually in a first step the decision making on the topology (i.e. the architectural design decisions) and the second step the dimensioning of the design parameters (i.e. the dimensioning of design components). 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 processing chains of engineering information occur 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 100 percent 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 known to be an important way to support collaboration. When well specified, 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 be able to support projects during their entire life-time. In the aerospace industry this implies a life-span of 50 years and 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 quite 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 since. 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 all others and to 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. Other domains, such as wire harness, undergo an similarly radical transformation process as product geometry has underwent over the last 30 years of CAD systems, but in much less time. As a consequence, current standards for harness information such as the VEC (Vehicle Electrical Container)[1] as the successor of the KBL[2] 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.[3]

All-in-all, it can be concluded that standards are potentially valuable, but they currently suffer from certain drawbacks 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. 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 almost timeless, enduring standard which would be complete and consistent and therefore a worthwhile financial investment into valid and secure digital process chains.

## 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 life-cycle.

Graph-based design languages all by themselves are a novel 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. Through the automatic compilation of a graph-based design language in a machine called design compiler, a powerful framework for engineering design can

[1] http://ecad-wiki.prostep.org/doku.php?id=specifications:vec:start
[2] http://ecad-wiki.prostep.org/doku.php?id=specifications:kbl
[3] https://www.vda.de/de/services/Publikationen/Publikation.~1025~.html$

be established. This relieves the design engineering teams by automatic model generation from tedious routine works, allows know-how re-use of design knowledge by re-use of design rules and eases topological and parametrical product variations.

In order to fully automate, semi-automatically or interactively assist such design and manufacturing development activities and processes along the product development process, novel means to represent the design and manufacturing knowledge needs to be developed. The so-called design language workbench on the basis of either the so-called "graph-based design languages", CPACS, or STEP is intended to solve several important issues in this respect:

- Representing engineering knowledge in a human-readable and digitally processable way according to the philosophical approach "design as a language", as it is described in the book by Brian Arthur ("The Nature of Technology - What It Is and How It Evolves", New York, Free Press, 2009).
- Decomposition and structuring of the engineering design knowledge in the form of a design language with a vocabulary (i.e. building blocks), rules (i.e. building knowledge) and process knowledge (i.e. building sequences)
- Allowing the merging, mapping and extension of the knowledge representation in form of design languages by processing mechanisms ensuring consistency and correctness.
- Model generation of all necessary disciplinary engineering analysis models by compilation of the design language into consistent, domain-specific model representations.

The sought-after engineering design language workbench 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. For this purpose, the concept and representation format of so-called graph-based design languages on the basis of the Unified Modeling Language (UML) will be used and partially extended. Since the creation of such an equally global generic and locally 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 design language workbench involves the following list of syntactical features definitions and developments:

- Demonstration of automated 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 automated 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 by means 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 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 design language workbench involves the following list of semantical feature definitions and developments:

- Abstract geometry ontology representation. This involves the definition of: geometry 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. This includes demonstration of extension capabilities for new geometry features. These features allows 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 in 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 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) and finite difference meshing schemes (CFD for fluid mechanics analysis).
- Validation of mesh enrichment with physical properties in a FEM-analysis (in the 1-month rudder use-case) and a CFD-analysis (in the 3-weeks cockpit use-case) 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 and the CFD-analysis with known reference analytical results (i.e. systematic testing).

For the listed development goals, IDEaliSM will make use of open, internationally standardized and IP free 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 future-proof 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).

The consortium sees a serious chance that a) the design language representation on the basis of UML will allow the representation of the design knowledge in an international, open source format independent from a special software vendor company, and, b) that the ontology mapping capabilities behind the design language workbench will provide an elegant, alternative way out of the dilemma to favour one standard over all others by providing appropriate translation means between the already existing individual domain standards as this will be illustrated later via the mapping between design languages based on UML, STEP and commonly used formats like CPACS and WFXML.

Since many standards define a data model and a XML schema for the exchange of data between OEM and supplier there exists a seamless way of integration with design languages based on UML, by means of style-sheet translation technology.

# 3 Task breakdown

## 3.1 T4.1: Design Language Workbench

The task of creating a Design Language Workbench will be entirely accomplished if methods could be defined and applied for decomposing and structuring engineering design knowledge in form of design languages, with a vocabulary (i.e. building blocks), rules (i.e. building knowledge) and process knowledge (i.e. building sequence).

Design Language definitions, which are developed in T4.2, 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".

## 3.2 T4.2: Domain Specific Languages (DSLs)

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. The resulting Domain Specific Languages 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. Abstraction of geometry will be a separate design language. This 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 generic design language components (vocabulary, rules and production systems) 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 design language for the modelling of use case 2 (harness in 10 days) and use case 3 (3 weeks cockpit) which can interact with other design languages which express other engineering design tasks.
- Implementation of an interface between the design language workbench and 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.
- Extension of the language workbench to facilitate a generic representation for future manufacturing means and processes.
- Standardized language to express business and simulation workflows

## 3.3 T4.3: Engineering Library

The development of an engineering library will take place to rapidly frontload engineering programs based on corporate standards.

The library will be composed of the following main features:

- process modules (tasks, deliverables, workflows, human-oriented and simulation-oriented)
- product modules (parts, assemblies)
- design requirements
- rules and constraints
- COTS (design) tools
- engineering services developed in WP3

Interface standards will be developed allowing quick and smooth integration of engineering modules into the appropriate programs.

The ontologies emerged from the Design Language Workbench will be used to define these interface specifications, since naturally a lot of variations in languages and structures among the different (types of) standards exist.

## 3.4 T4.4: Standard Interfaces and Exchange Formats

In this task data formats and interfaces have to be established which represent the projects knowledge in an integrated manner.

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 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
- the integration and management of such information from different sources in a consistent database
- the 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.

## 3.5 T4.5: Modelling of Cable Harnesses

This task concerns the development of a specific solution 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. A semi-automated software tool will be used for the prediction of cable harness stiffness and results will be validated with experimentally measured data (this links to WP5).

This task contributes to Design Languages (T4.2) and is a part of the Engineering Library (T4.3) as a tool to calculate harness stiffness data.

# 4 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.

## 4.1 Fraunhofer LBF

| | |
|---|---|
| **Software application name (version)** | LBF-CHSSC (Cable Harness Segments Stiffness Calculator) |
| **Engineering services provided** | Stiffness Calculation of Cable Harness Segments |
| **Operating system (version)** | Microsoft Windows 7<br>Java Runtime Environment 8<br>Ansys (R14.5, R15.0)<br>Screen resolution > 1200 x 850 pixel |
| **Virtual machine support (version)** | No? |
| **Data formats support (version)** | tbd (KBL, VEC support planned, maybe also XML, STEP or CPACS support useful) |
| **Information model availability, name (version)** | ? |
| **Information modelling language to document the information model** | tbd |
| **Programming languages support** | tbd |
| **API support** | tbd |
| **Web-services support** | ? |
| **Provided test data** | tbd |
| **Contact name (email address)** | Christoph Tamm (christoph.tamm@lbf.fraunhofer.de) |
| **Other information** | |

## 4.2 IILS

| | |
|---|---|
| **Software application name (version)** | DesignCompiler43 (version 2.1) |
| **Engineering services provided** | 3D cable routing |
| **Operating system (version)** | Windows, Linux (no special version)<br>64-bit recommended |
| **Virtual machine support (version)** | with client operating system Windows or Linux |
| **Data formats support (version)** | datasets: *.xls<br>electrical information: *.kbl<br>geometrical information: *.step, *.stl, *.vtp |
| **Information model availability, name (version)** | own data model 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** | |

## 4.3 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)<br>Bert Lagaisse (bert.lagaisse@kuleuven.be)<br>Bart van Brabant (bart.vanbrabant@cs.kuleuven.be) |
| **Other information** | https://github.com/impera-io/impera<br>Support for deploying on OpenStack (private cloud) and Amazon AWS |

## 4.4 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** | |

## 4.5 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)** | Maarten Nelissen (maarten.nelissen@ke-works.com) |
| **Other information** | |

## 4.6 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.<br>libraries to connect analysis modules to the central data model CPACS |
| **Operating system (version)** | Red Hat Enterprise Linux 6 Workstation (64 bit)<br>Debian 7 stable (64 bit)<br>SUSE Linux Enterprise Desktop ("SLED") 11 SP2 (64 bit)<br>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<br>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 address)** | Erwin Moerland (erwin.moerland@dlr.de),<br>Thomas Zill (thomas.zill@dlr.de) |

| | |
|---|---|
| **Other information** | Contact persons of DLR's software department:<br>Doreen Seider (doreen.seider@dlr.de),<br>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<br>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),<br>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 |

## 4.7 Kontec

| | |
|---|---|
| **Software application name (version)** | DesignCompiler43 (version 2.1) Plugin |
| **Engineering services provided** | Interface for FEM simulation in Compiler 43<br>Engine & Exhaust aftertreatment development |
| **Operating system (version)** | Windows 7 |
| **Virtual machine support (version)** | ? |
| **Data formats support (version)** | datasets: .xlsx, .txt<br>meshing information: .unv (GMSH), .inp (Abaqus)<br>geometrical information: .stp, .stl, .step<br>FEM results: .frd (Calculix) |
| **Information model availability, name (version)** | own data model based on UML |
| **Information modelling language to document the information model** | UML |
| **Programming languages support** | Java, VBA |
| **API support** | No / not yet |
| **Web-services support** | No / not yet |
| **Provided test data** | none |
| **Contact name (email address)** | Qi Xie (qi.xie.dif@kontec.de) |
| **Other information** | |

## 4.8 NOESIS Solutions

| | |
|---|---|
| **Software application name (version)** | Noesis Optimus 10.16 and higher<br>Noesis Optimus 11 enterprise 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.<br>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)<br>Windows Vista on x86 and x86-64 (both AMD & Intel hardware)<br>Windows Server 2008 on x86 and x86-64 (both AMD & Intel hardware)<br>Windows 7 on x86 and x86-64 (both AMD & Intel hardware)<br>Windows 8/8.1 on x86 and x86-64 (both AMD & Intel hardware)<br>Linux SUSE Enterprise 10.3 and higher on x86 and x86-64 (native 64-bit supported)<br>Linux RedHat Enterprise 5, 6 and 7 on x86 and x86-64 (native 64-bit supported)<br>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** | Worfklow 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.<br>No REST server yet implemented (planned) |
| **Provided test data** | none |
| **Contact name (email address)** | Roberto d'Ippolito (roberto.dippolito@noesissolutions.com) |
| **Other information** | |

# 5 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 might be used as interchange formats. This section starts with a description of the master data management (MDM) module, performing the central management of data within the advanced engineering framework.

## 5.1 Standardisation Strategy

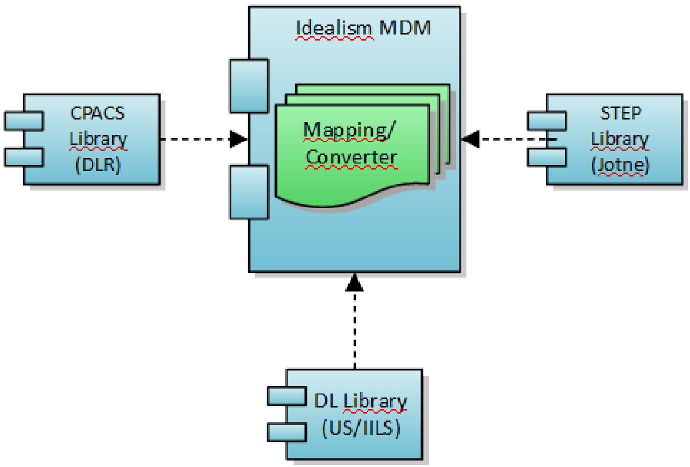


**Figure 2: Data Formats contributing to the MDM**

In order to guarantee consistent data management in WP4, a master data management (MDM) module will be established, depicted in Figure 2. This MDM is capable of providing data to the other modules within the Advanced Integration Framework, 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 will be established aiding in the translation from the one to the other. For the data exchange between the standard data formats and the workbench functionalities (e.g. FEM, Routing …), the ontologies of corresponding design languages act as interfaces (developed in T4.2). These ontologies are published and maintained by the design language developers and integrated into the standard data formats of the master data model.

## 5.2 BPMN

The Business Process Modelling 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. its 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.

## 5.3 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).

## 5.4 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

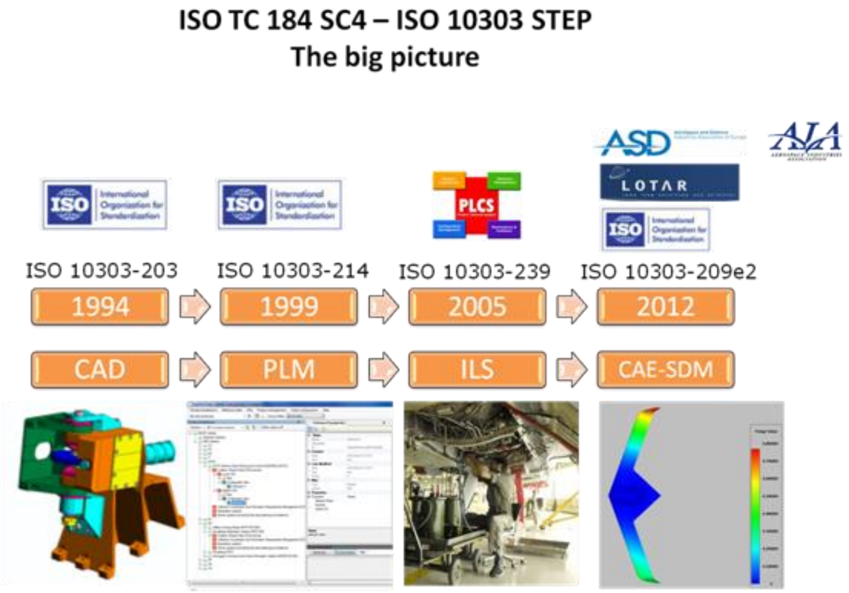


**Figure 3: The development of the STEP standard over the years**

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.

# 6 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. These common information formats are often based on XML (see annex).

## 6.1 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.

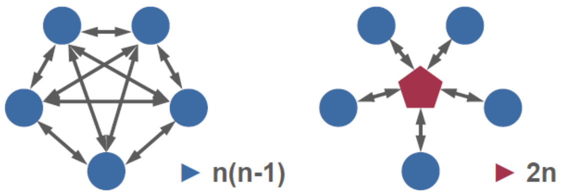


**Figure 4: A Central Model Approach significantly reduces the amount of interfaces within a design process**

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.

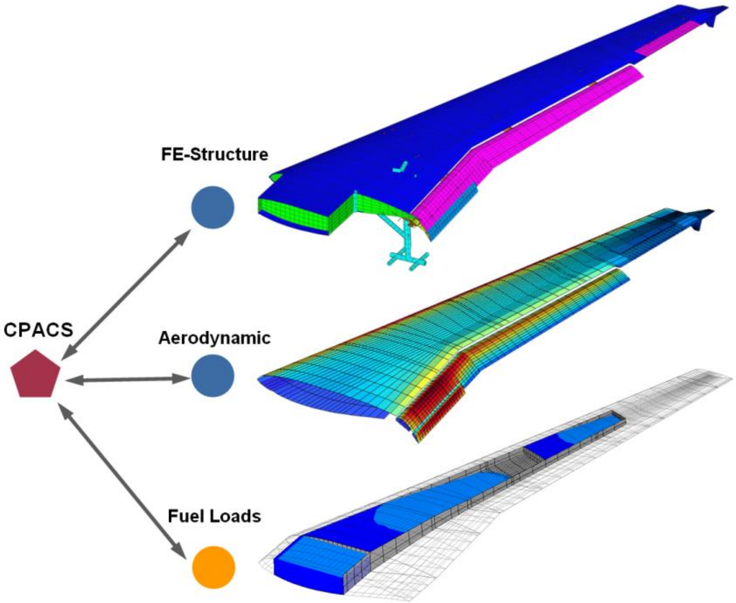


**Figure 5: Example of multi-disciplinary analysis using CPACS**

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/.

## 6.2 Graph-based design languages based on UML

Graph-based design languages are a novel 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. Graph-based design languages are expressed on the basis of the internationally standardized Unified Modeling Language (UML) format and are therefore easily readable, editable and storable based on publicly available UML tools.

# 7 Conclusion

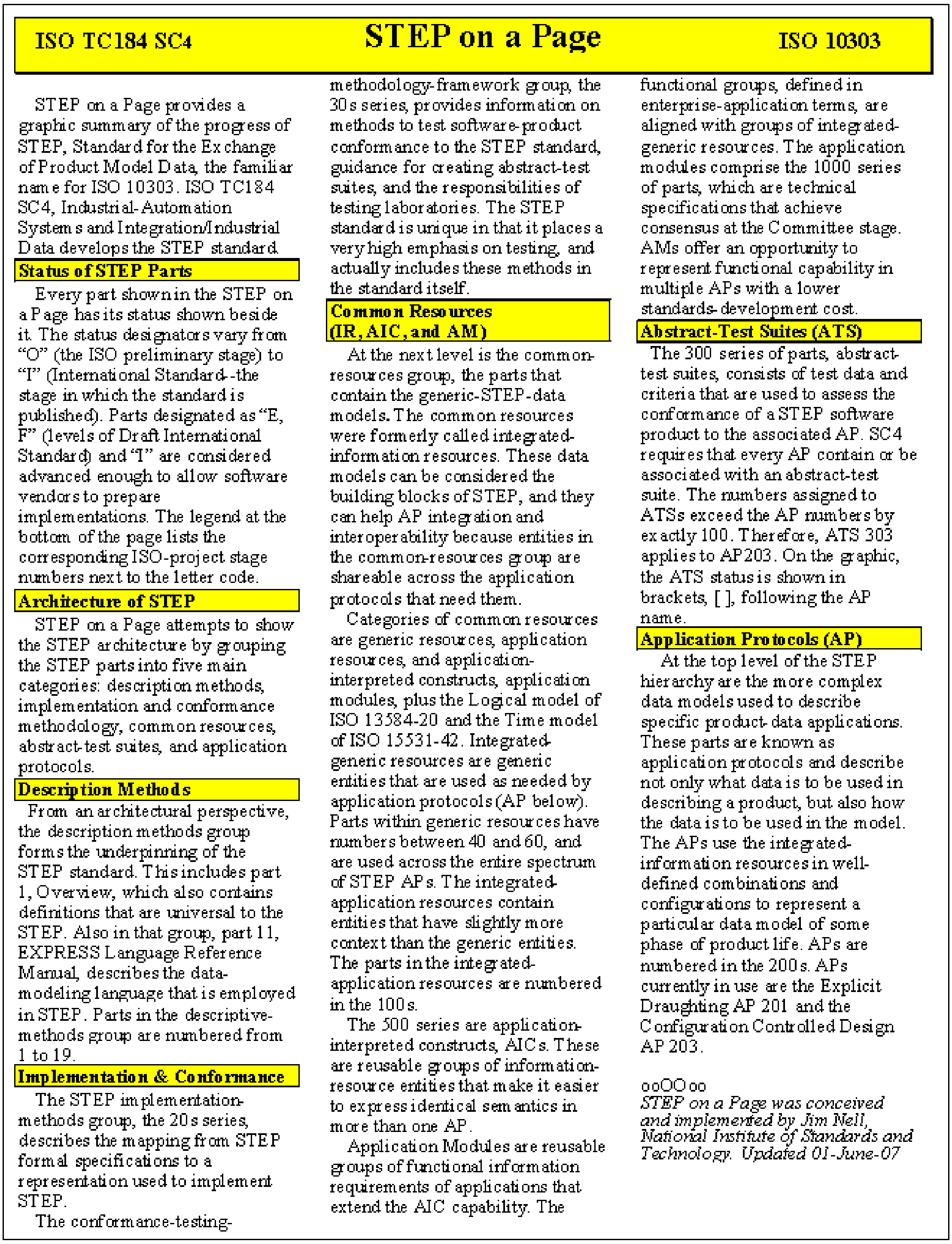
This document created a foundation for the creation of tool interconnectivity and interoperability within the IDEaliSM project. Section 3 provided an overview of the tools that will be incorporated in the project by the solution providers. The list of supported interfaces and formats defines the baseline for data exchange. Section 4 provided an overview of the intended usage of data formats in the Master Data Management module within the Advanced Integration Framework as developed in Work Package 3 of the project.

At this stage of the project the use-cases are not yet fully defined. The types of information required to work out the use-cases can still vary during the course of the project. Ontologies describing the information have to be defined and integrated into a common information model like CPACS or STEP as well as in graph-based design languages based on UML.

Also depending on in-house processes of different companies it is likely necessary to handle different Domain Specific Languages and data formats beside the currently supported ones (section 3) and proposed in section 4. At a later stage, the required information could be incorporated into the integrated data formats or alternatively appropriate converters have to be created where necessary.

# 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)**

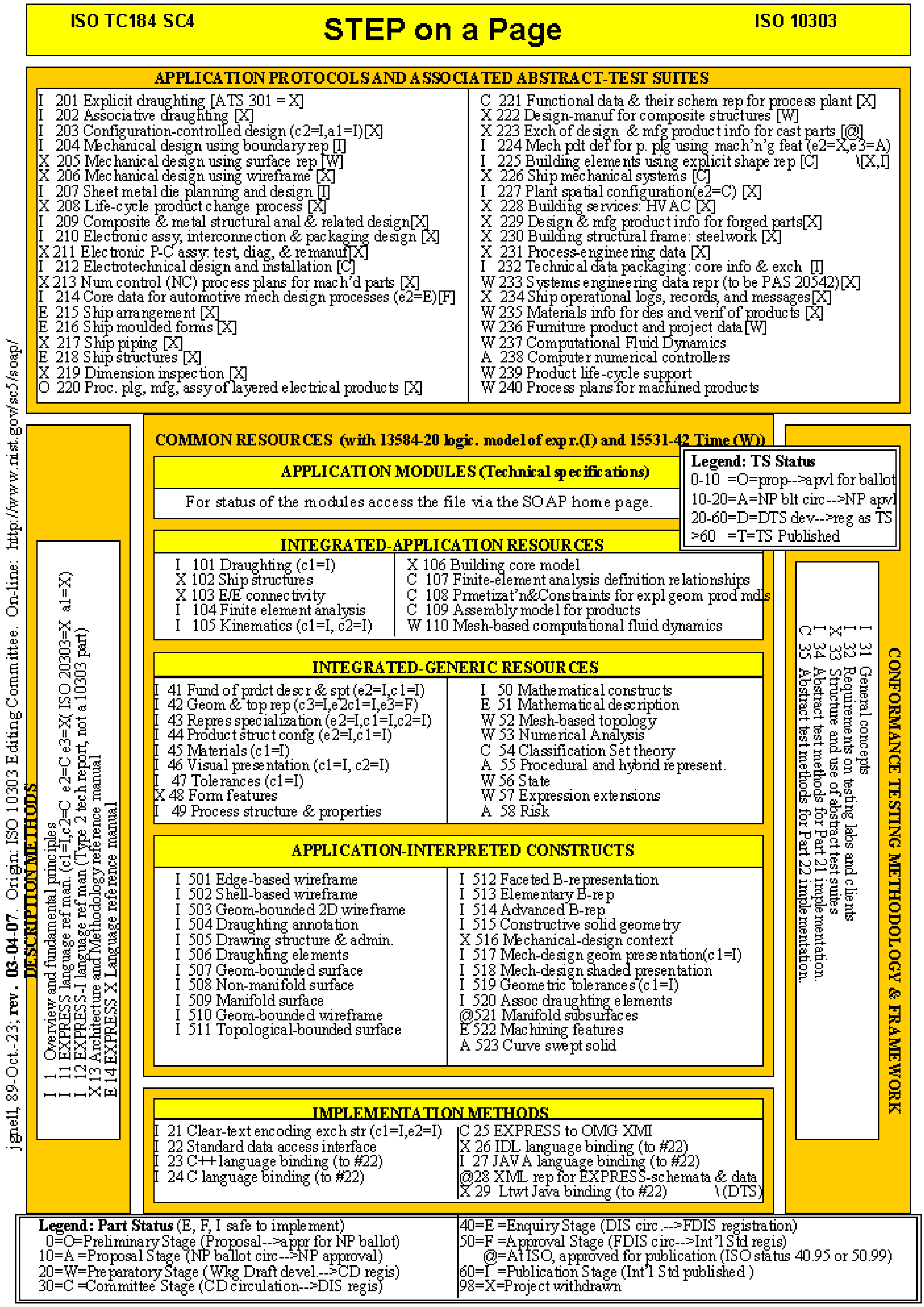
The 300 series of parts, abstract-test suites, consists of test data and criteria that are used to assess the conformance of a STEP software product to the associated AP. SC4 requires that every AP contain or be associated with an abstract-test suite. The numbers assigned to ATSs exceed the AP numbers 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)

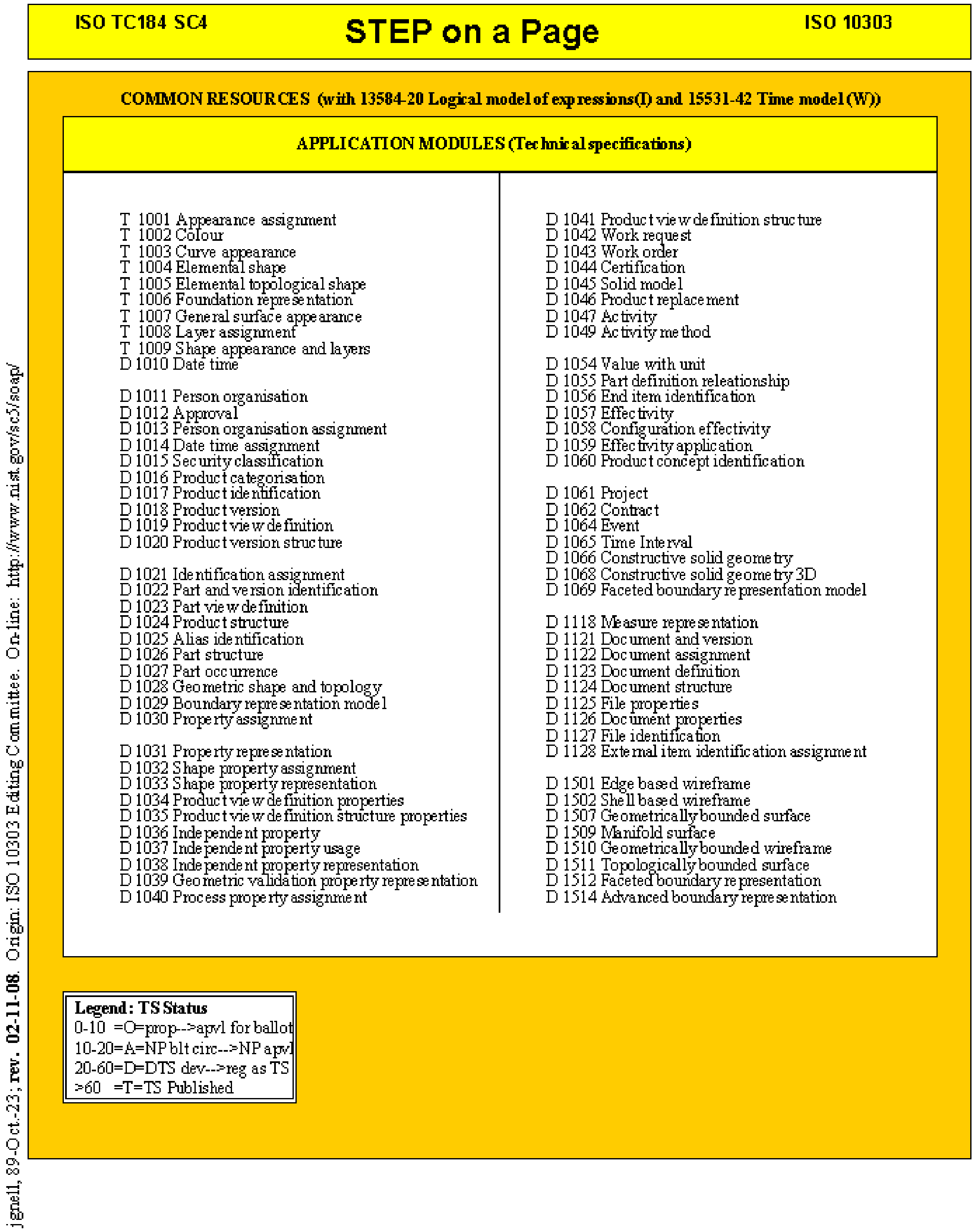
C 25 EXPRESS to OMG XMI
X 26 IDL language binding (to #22)
I 27 JAVA language binding (to #22)
@28 XML rep for EXPRESS-schemata & data
X 29 Ltwt Java binding (to #22) \(DTS)

**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.