



How to Advance Interdisciplinary Model Based Engineering of Space Systems?

Hans Peter de Koning (ESA)

SESP, 28-30 March 2017, ESA/ESTEC

ESA UNCLASSIFIED – Releasable to the Public



European Space Agency

Sharing Information Between Disciplines



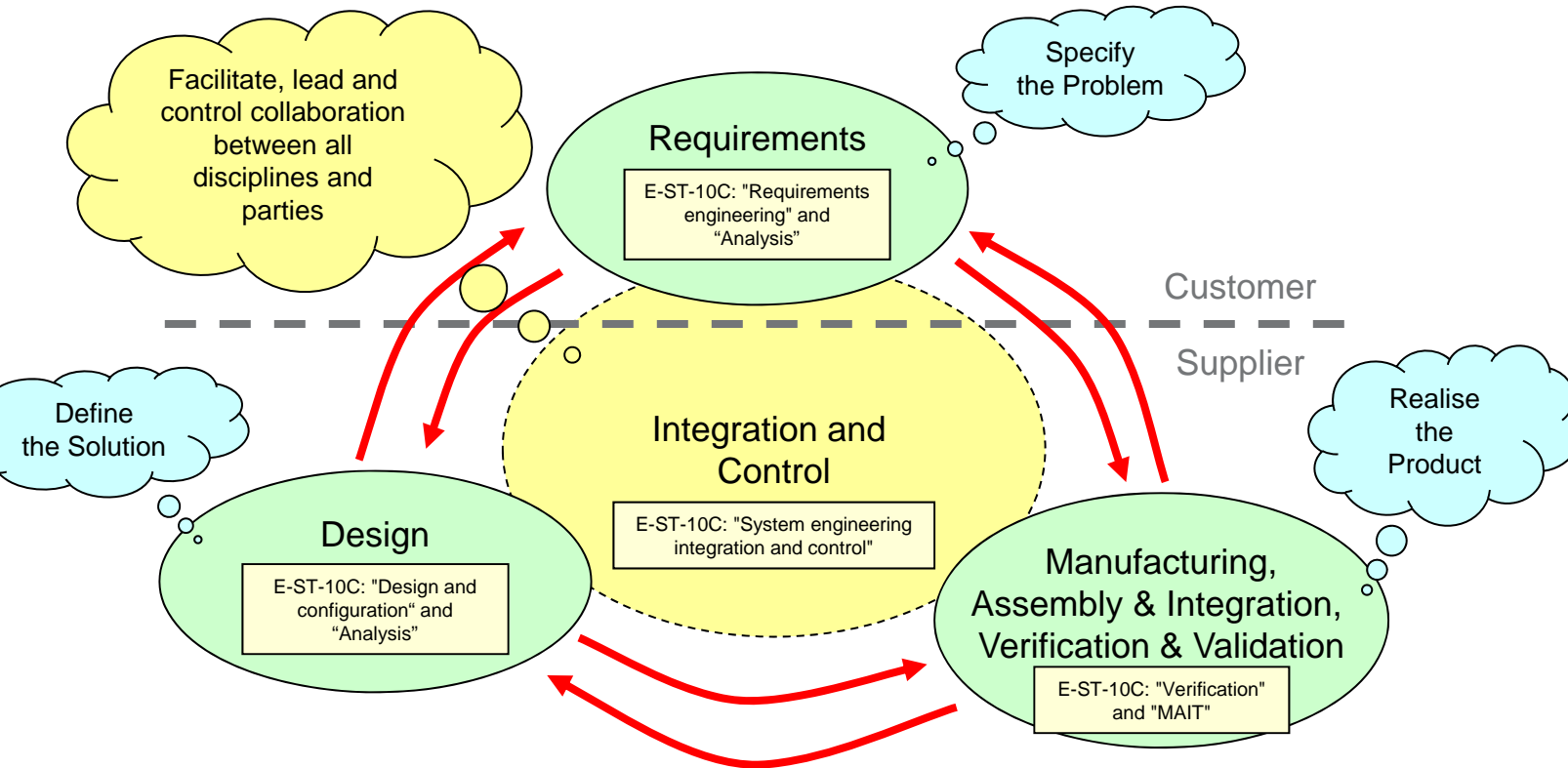
*getting the right version
the right information
the right team member
the right time ...*

*providing consistent, complete, navigable, reviewable information
while making a deadline*

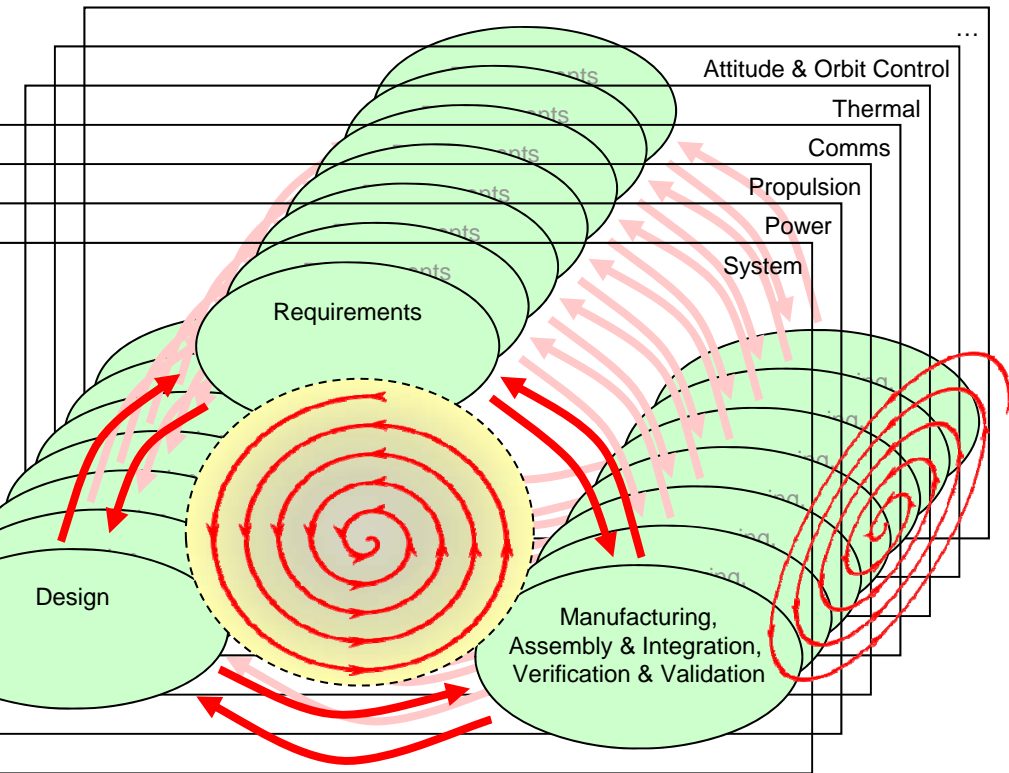
major challenges in all our projects

Digital engineering / model based approaches promise substantial
improvements ... but do not fall from the sky for free

System Engineering according to ECSS-E-ST-10C

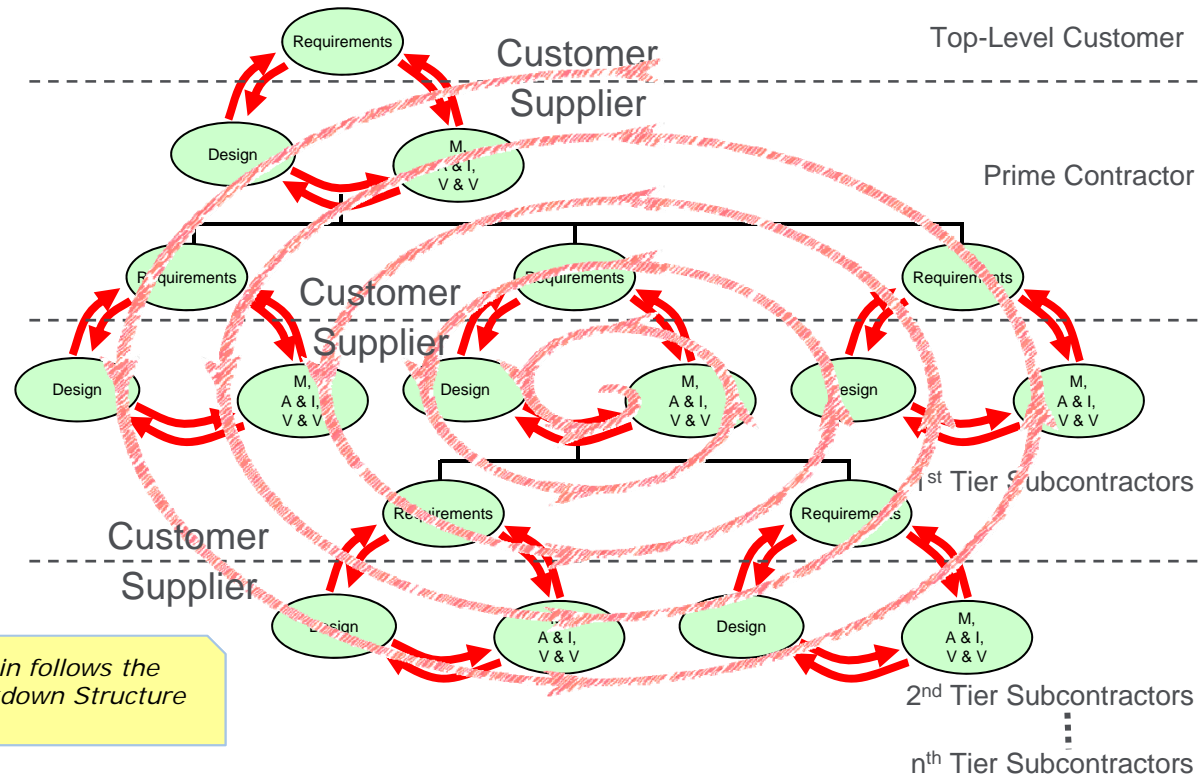


S-E-ST-10C: Iterative “Integration and Control”

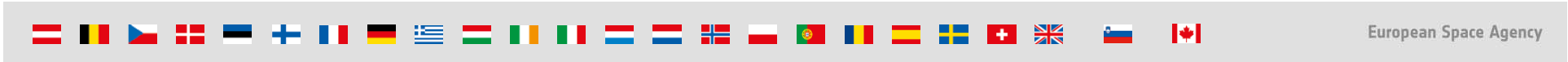


- Integration and Control
 - ❖ Concurrent in early phases
- Iterate between Requirements, Design, and MAIV&V
- Iterate across Disciplines

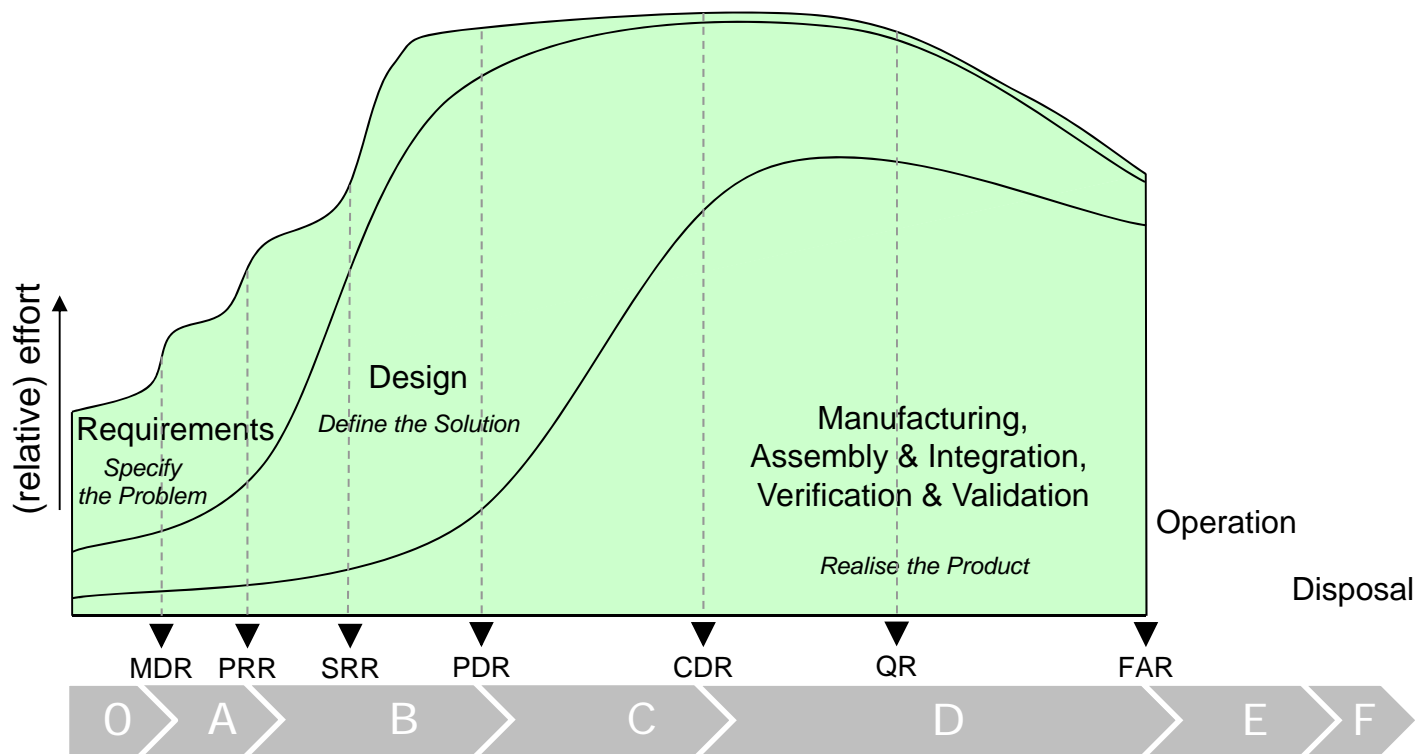
ECSS-E-ST-10C: "Integration and Control" Across the Customer-Supplier Chain



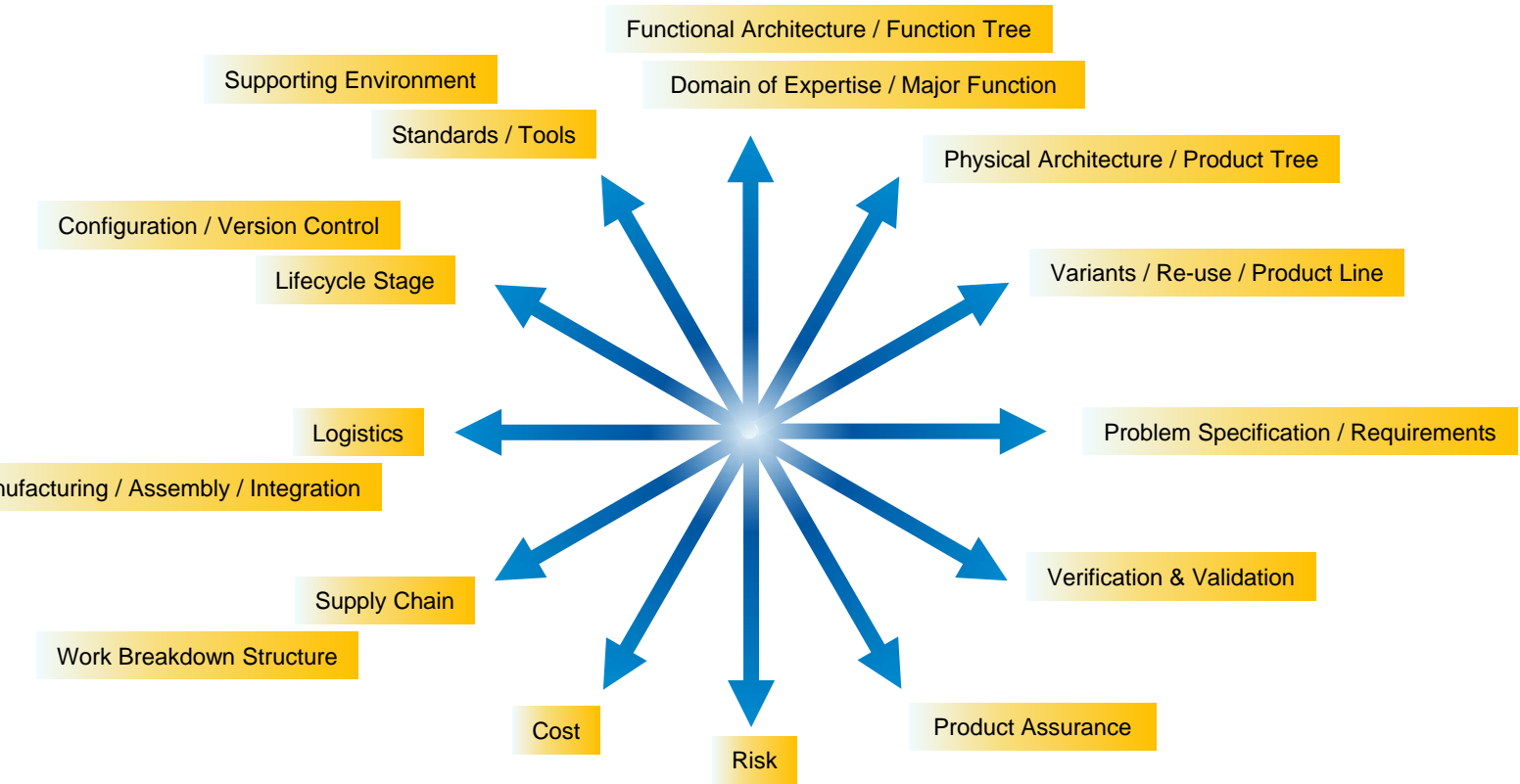
The Customer-Supplier Chain follows the Product Tree or Work Breakdown Structure or a combination thereof



S-E-ST-10C: "Integration and Control" effort along the System Life-Cycle



Large Information Management / Knowledge Representation Exercise ... across many dimensions



Need Rigorous System Engineering



Very inspiring talk at NASA/JPL
MBSE Symposium – Jan 2017 –
Steven Jenkins (JPL)

“Systems Engineering Really
Engineering?”

Rhetorical, rather: “How do we
ensure that systems engineering
really is engineering?”



National Aeronautics and
Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

A Few Words About Rigor

- **Rigor in engineering is a distinguishing virtue**
 - it's what we do
- **Rigor requires no justification and we offer none**
- **Rigorous does not mean *detailed***
 - it means simplification must be justified
- **Rigor is not a value to be traded against time or money**
- **Rigor benefits all endeavors, simple and complex**
- **Rigor benefits all projects, large and small**
- **Rigor leads to**
 - better understanding of mission objectives and constraints
 - more precise descriptions of design concepts and realizations
 - more thorough and principled verification and validation
 - earlier and more effective remediation of defects
 - more accurate projections of budget and schedule

2017-01-25 JPL MBSE Symposium 4



Notes from Steven Jenkins



National Aeronautics and Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

How Can We Do Better?

exploit graph theory: the mathematical study of graphs, which present pairwise relations between objects

- Knowledge representation theory makes heavy use of graphs
- There is a natural connection between description and analysis

use graph theory to structure and organize the facts (language assertions) about the objects of our design and analysis

- We can reason about whether the resulting graph is well-formed according to the rules of our language
- We can reason about all kinds and degrees of relatedness
 - e.g., What requirements does this requirement directly refine?
 - Indirectly?

exploit well-known graph algorithms for engineering problems

- connected components: fault isolation
- transitive closure: state reachability
- topological sort: root cause analysis

JPL MBSE Symposium 14

National Aeronautics and Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Conclusion

- Systems Engineering is really Engineering to the degree that it achieves rigor through in description and analysis through**
 - precise language with rules and meaning
 - mathematical abstractions
 - automation
- Graph theory is a fundamentally applicable abstraction that empowers both description and analysis**
- I don't like the term *Model-Based Systems Engineering* because it leads to silly questions like "What is a Model?"**
- But I would *describe* MBSE as Systems Engineering practice that achieves rigor through use of**
 - precise language for description
 - mathematical abstractions for analysis
 - effective automation

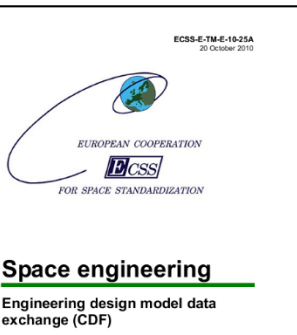
2017-01-25 JPL MBSE Symposium 17



S Semantic Data Models in support of MBSE

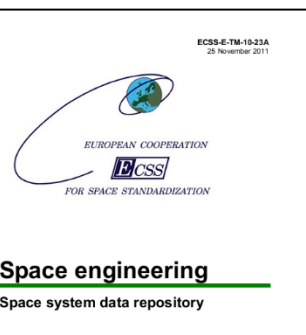


E-TM-10-25



- Focuses on conceptual models in early life cycle phases (O, A)
- Implemented in OCDT

E-TM-10-23



- Focuses on detailed models in later life cycle phases (B, C, D, E)
- Implemented in VSD

- Developed in tandem
- Where possible common approach and semantics
- Where possible aligned with OMG SysML
- Plan: Future merge to single real standard
- E-TM-10-25A made available Oct 2010
- E-TM-10-23A made available Nov 2011



M-10-23 & 25 Approach



was realised early on: in order to ensure long term interoperability must create semantic conceptual data model – i.e. the ontology approach

At the time (2006-2011) not yet the means nor expertise ...

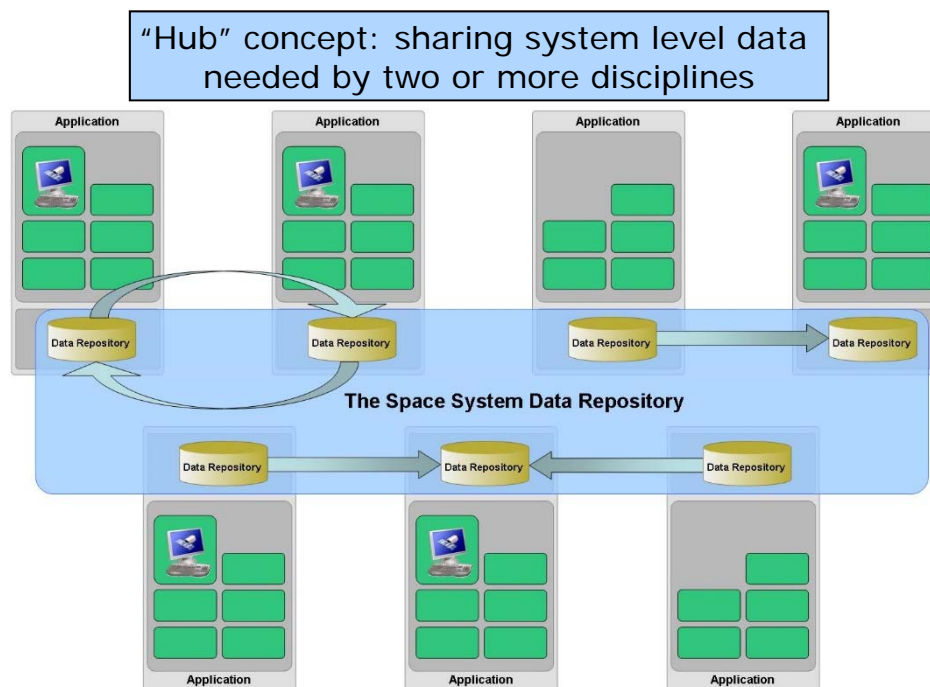
... Best effort with 'semantic' UML / Ecore models Auto-generate implementation technology from conceptual data model – as much as possible

Information sharing via "Hub": Space System Data Repository

Federation of data stores with adapters complying to semantic standard model

Approach reconfirmed in 2014 Technical Harmonisation "Space System Data Repository"

"Hub" concept: sharing system level data needed by two or more disciplines



Patterns implemented in E-TM-10-25 (and OCDT)



Ownership / responsibility by Domain of Expertise

“Domain of Expertise” is generalization of “Discipline”

Unambiguous stable object identifiers – UUIDs

Separation of “Core Data Concepts” and “Reference Data”

Core Data Concepts are hard-coded in (generated) software implementation

Reference Data is loaded at run-time and provides extension mechanism

Rigorous formal model of Quantities, Units, Scales, Physical Dimensions

Web Service with simple HTTP(S) REST API

Encapsulates persistent data store / hides implementation detail

Compatible with secure traversal of corporate firewalls



Ownership / Responsibility by Domain of Expertise

Person represents a user
with **PersonRole** & **PersonPermission**
at **SiteDirectory** level

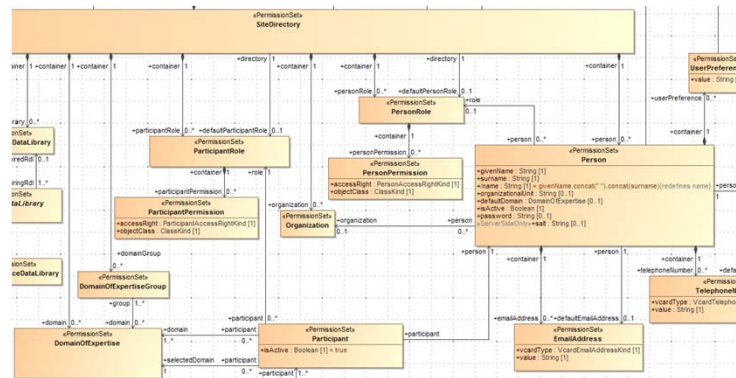
Participant is a **Person** representing
one (or more) **DomainOfExpertise**
in one **EngineeringModel**

with **ParticipantRole** & **ParticipantPermission**
at **EngineeringModel** level

Participant acts as one **DomainOfExpertise** at any time in a session

every model **Element** and every **Parameter** in an **EngineeringModel** is owned
by one **DomainOfExpertise** who is responsible for its definition / value

non-owner **DomainOfExpertise** can take a **Subscription** on **Parameter** to
use it as input



Ambiguous Stable Object Identifiers – UUIDs



UUID version 4 as object id on all classes

No central id authority needed

Solid algorithms available in all programming languages

Allows renaming of human readable identifiers without need for schema migration

No collisions in 2 years of operation in CDF

Simplifies primary / foreign keys in persistent and in-memory data stores

ExternalIdentifierMap to capture correspondence mapping between

TM-10-25 / OCDT UUIDs and identifiers in external models

Reduces as much as possible the loss of information when performing round trip

import / export data transfer between an E-TM-10-25 compliant model and a

model in the format of an external tool

Evolution of “Core Data Concepts” “Reference Data”



ReferenceDataLibrary (RDL)

ParameterType definition of **Parameter** type (everything except its value)

- Comprises **Text**, **Date**, **Time**, **Boolean**, **Enum**, **QuantityKind** scalar subtypes, as well as **CompoundParameterType**

Category for user-defined categorization / filtering of concepts

Rule for user-defined verification rules

MeasurementUnit and **MeasurementScale** to support **QuantityKind**

- Same concepts as SysML QUDV model library
- Establishes all info needed for automated unit/scale value conversion
- Supports ratio, interval, logarithmic, cyclic and ordinal scales

Constant for mathematical, physical, model constants

Glossary of Terms

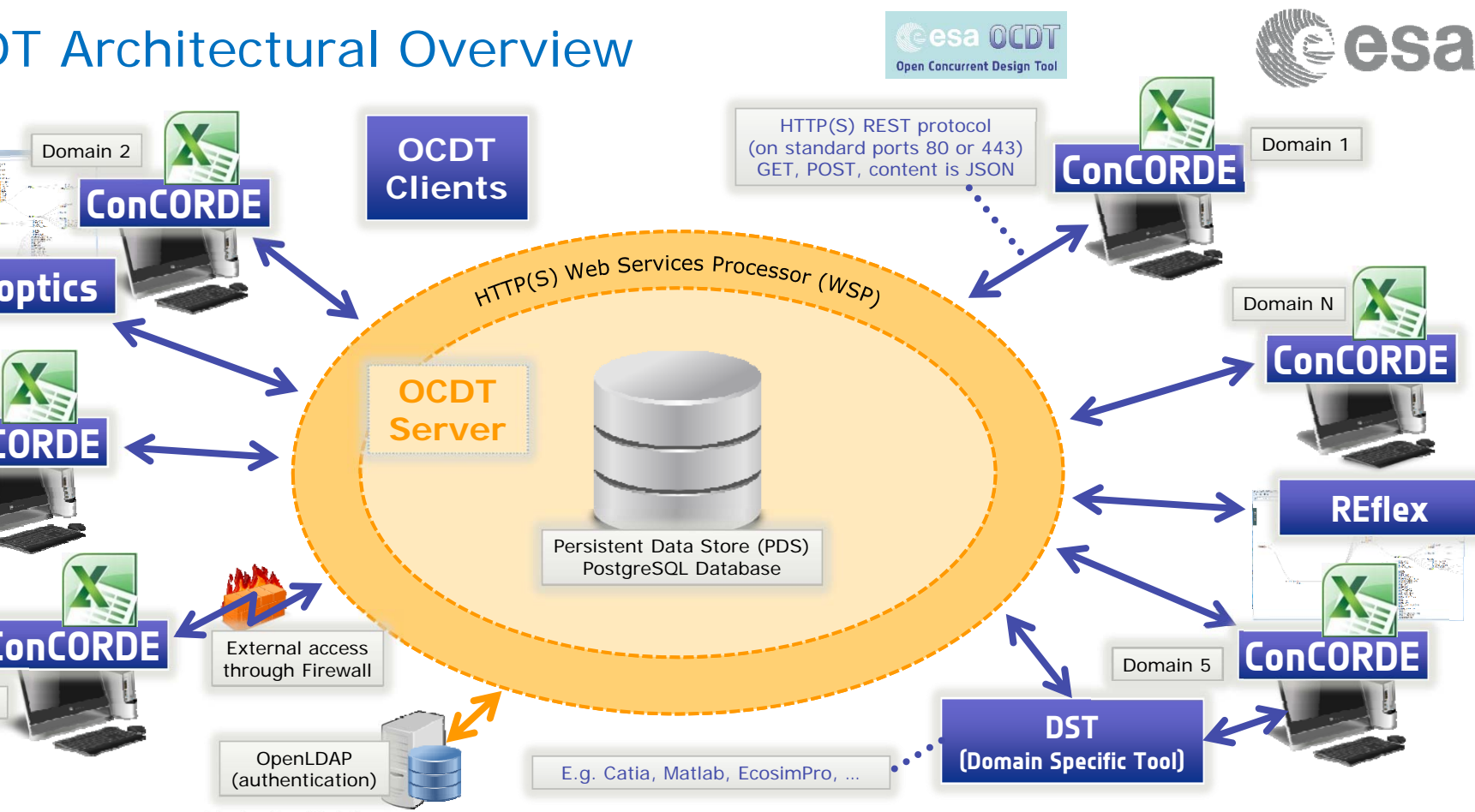
ReferenceSource and **Citation**

Generic RDL provided by ESA defines **ParameterTypes** most used in space missions

Including ISO/IEC 80000 quantities, units, scales

Multiple **RDLs** may be chained: e.g. model-specific → family-of-projects → generic

OCDT Architectural Overview



Service with simple HTTP(S) REST API



Standardized in E-TM-10-25 Annex C

See <https://ocdt.esa.int/projects/ocdt/wiki/ECSS-E-TM-10-25-Annex-C>

HTTP(S) GET and POST on standard ports 80 or 443

JSON request / response body – flat array of JSON serialized objects

ACID safe transactions with PostgreSQL database backend

URI navigation follows top-down composite structure of conceptual data model

Query parameter *extent=shallow/deep* to get single object or object (sub)graph

Supports getting delta since given revision number

Compatible with near-real-time support for 50+ concurrent users and multiple models at 30 seconds synchronization interval for all users

Additional off-line exchange file format in similar JSON files in ZIP archive

OCDT implementation on nodejs in TypeScript

Very positive experience – robust, flexible, performant, easy to debug

Main Specific Tool Integrations



- “Quick and Dirty” integrations via Excel worksheets
- Catia v5 bi-directional interface for 3D Configuration in CDF studies
 - Basic geometric shapes and coordinate transformations
 - Catia computes centre-of-gravity and moments-of-inertia
 - Currently alpha version – full operational release expected summer 2017
- Bi-directional SysML / UPDM interface (MagicDraw / ESA-AF)
 - Alpha version developed in CESoS activity (2014)
- Cycle Assessment tool OPERA (in support of CleanSat)
 - Expected operational release summer 2017
- Matlab interface by University of Madrid
 - Alpha version demonstrated in SECESA 2016
- Requirements Engineering – DOORS via ReqIF, and SysML (MagicDraw)
 - In progress in Flexible Wiki-based Requirements Engineering activity: REflex
 - Expected operational release summer 2017
- Handover to VSD successfully prototyped, connection with e.g. MARVL in future
- Interfaces for Capella, EcosimPro, Thermal via STEP-TAS, maturing SysML and Matlab interfaces, Ground Segment Engineering ...

Conclusions & Outlook



Experiences with E-TM-10-25 / OCDT show that “Multi-Disciplinary Hub” based on semantic conceptual data model starts to work

For Phase 0 / A / B type data

Feature requests / improvements collected on OCDT Portal Backlog

Good reasons to continue with pragmatic incremental approach

Not forgetting long term goals

Feed back lessons learned into further ESA, ECSS and OMG SysML v2

Next SysML version 2 looks like taking same approach

REST-like services API will become part of the standard

Will most probably get much cleaner / ontology like meta-model allowing for Hub capability

Major emphasis on usability and reducing the learning curve

RFP expected Dec 2017 – Standard and implementation around 2019 / 2020

Continue and deepen semantic modelling approaches based on formal logic, semantic web technology (RDF/OWL/Open Linked Data, FBM) including automated reasoning

Continue operations in ESA CDF and with ESA partners in industry and academia

OCDT Community Portal



The screenshot shows the OCDT Wiki page on the ESA Extranet. The page title is "OCDT Wiki" and it provides information about the portal's purpose, community access, and the Open Concurrent Design Tool. A diagram below the text illustrates the system architecture, showing the interaction between the OCDT Server and OCDT Clients across multiple domains.

OCDT Wiki

The purpose of this portal is to share and exchange information, experiences, models and tools related to concurrent engineering for space applications. In particular, the portal is dedicated to the development, distribution and usage of the **Open Concurrent Design Tool**.

Community and Access

In order to gain access you need to qualify as a member of the **OCDT Community**, see **What is the OCDT Community?**. A user account can be requested via e-mail to the portal administrators: hans-peter.de.koning@esa.int and sam.gereme@esa.int (for the time being, please send to both e-mail addresses).

What is the Open Concurrent Design Tool?

OCDT is an initiative of the **European Space Agency**, and in particular its **Concurrent Design Facility** at ESA/ESTEC, to create an open software environment for concurrent, collaborative and distributed engineering applied to the development of space systems in the early phases of the life cycle.

- It is an implementation of the draft open standard defined in the ECSS-E-7M-10-25 System Engineering – Engineering Design Model Data Exchange (EDM) Technical Memorandum.
- It has a three tier clients / web services / database server architecture and consists of the following main components (see also the Diagram below):
 - ConCORDE (Concepts, Options, Requirements and Design Editor), the main end-user client tool that is implemented as an add-in on top of Microsoft Excel® 2010 in the C# programming language.
 - Web Services Processor (WSP), implemented using the open source **nodejs** scalable network application platform, implemented in the CoffeeScript and Javascript programming languages.
 - Persistent Data Store (PDS), implemented using the open source **PostgreSQL** object-relational database management system, implemented in the SQL language (PostgreSQL version).
- It is a platform for exchange and near-real-time sharing of concurrent engineering models in order to enable distributed concurrent design sessions. All client-webservices communication uses the **HTTPS** protocol via standard ports (80 or 443) so that problems concerning inter-organisation links (through corporate firewalls) are minimised.
- In addition to the mentioned components there are also software development kits that provide libraries in different programming languages to support efficient development of other OCDT based/compliant applications and components.
- In the future there will also be domain specific tool adaptors that will integrate various Domain Specific Tools such as Caba, Matlab and EcosimPro as clients with the OCDT platform.

System Architecture Diagram:

- OCDT Server** components: PostgreSQL (Protocol over TCP/IP), Persistent Data Store (PostgreSQL RDBMS), Web Services Processor (nodejs on Google V8), Firewall (optional).
- OCDT Clients** components: ConCORDE (Domain 1, 2, 3), DST (Domain Specific Tool).
- Protocols:** http(s) REST protocol (on standard ports 80 or 443) JSON content.

<https://ocdt.esa.int>

<mailto:ocdt-support@esa.int>

