Gallery

arKItect is a platform enabling structuring, visualizing, and operating complex set of data stemming from various legacy sources (Excel, database, web sites, xml...) or edited directely through a collaborative man-machine interface.

arKItect modeler & database technology provides advanced support for handling hierarchies and interactions.

Because systems are made of (sub-)systems in interaction, our technology perfectly fits for complex systems modeling or organizing complex sets of data.

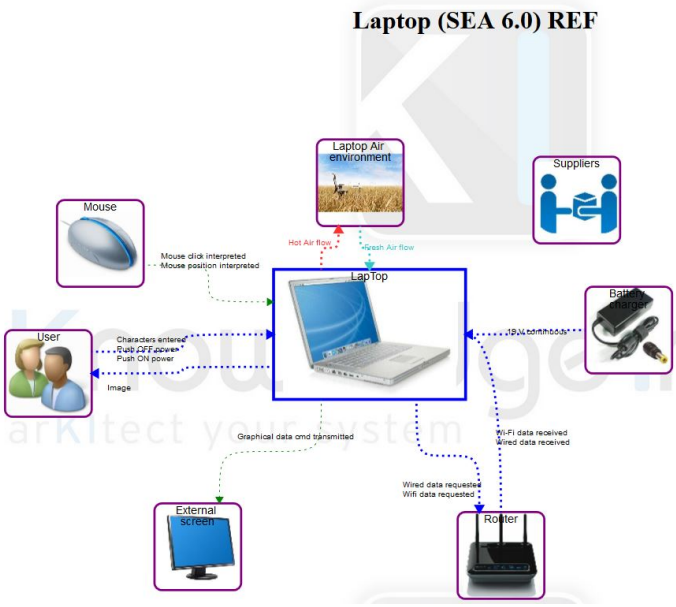
arKItect platform is general purpose and you can easily define your own data model, views, imports, exports, operating programs, editing modes, custom python tools.... However, we also offer custom configurations for **Systems Engineering** and **Project Development Plan** that have been developped with industry leaders (**Renault Nissan**, **Thalès Alenia Space**)

For additional information, refer to www.k-inside.com

Please CLICK on any of the images below in order to browse related system structure

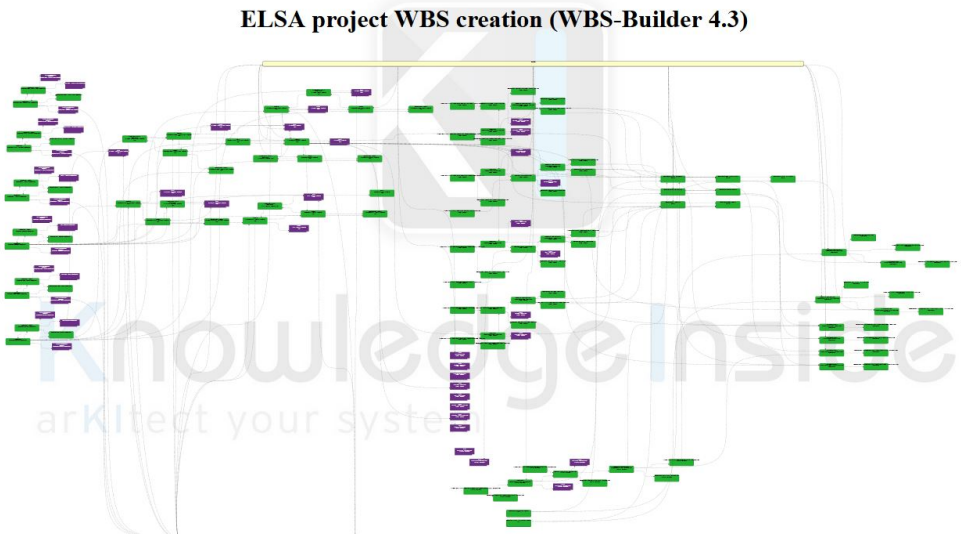
Please Use preferably Firefox or Edge for browsing

Systems Engineering Database and synchronized views - example of a Laptop



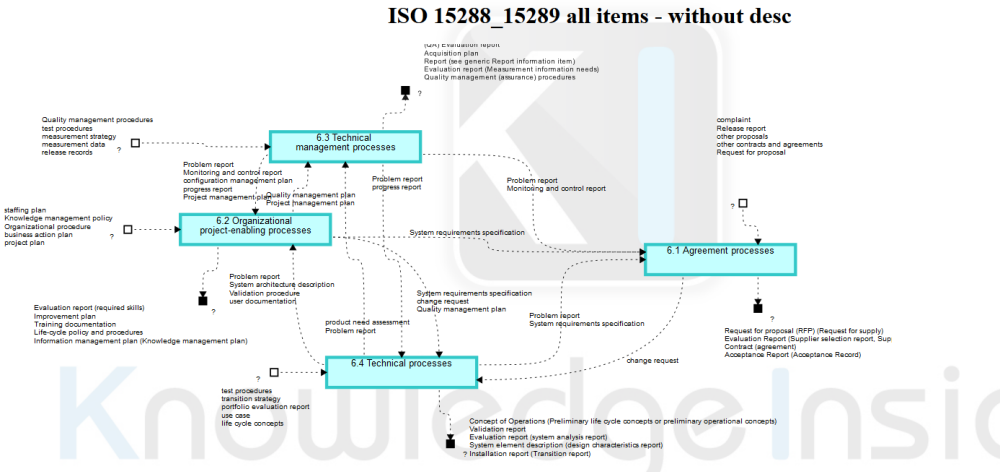
Here is an exemple of structuring and visualizing systems engineering data: requirements, functions, flows, components, interfaces,... all these artefacts are organized in through different hierarchical views that would typically fit with systems engineering processes: requirements management, functional and technical architecture.

Project Breakdown Structures (WBS, PBS, Processes...) structured and synchronized in an arKItect database



This is a PERT graph for a satellite project. Each green box is a workpackage, each violet box is a review. The objective behind structuring project data is to manage synchronously the Product Breakdown Structure, the Processes, the Work Breakdown Structure and the Organization Breakdown Structure. Normally, risks and planning may also be closely linked.

ISO 15288_15289

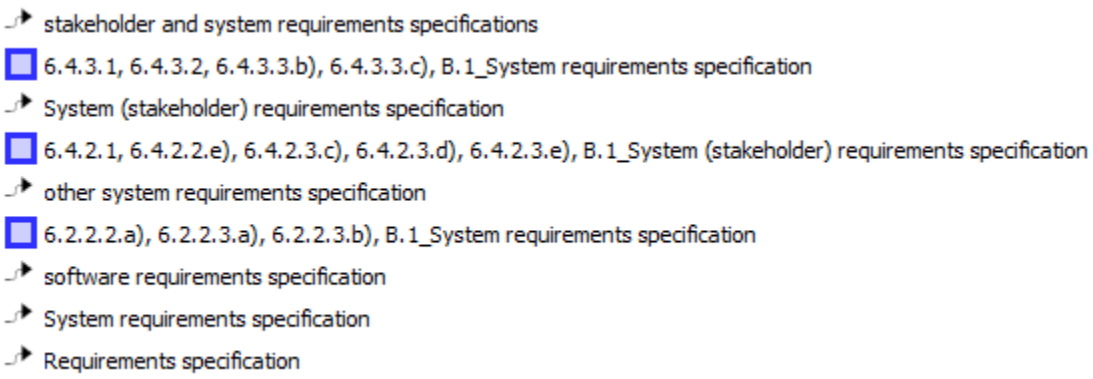


This web page is a model of ISO 15288:2015 together with ISO 15289:2017.
We started from the pdf document of the ISO and imported them in arKItect using a python API for reading pdf documents and importing them into an arKItect database.
Here is a data visualization of the arKItect database provided for free by arKItect platform from which we made an HTML export.

Conventions are as follows:
- cyan blue boxes are processes according to ISO 15288
- leaf blue boxes are activities which means parts of processes.

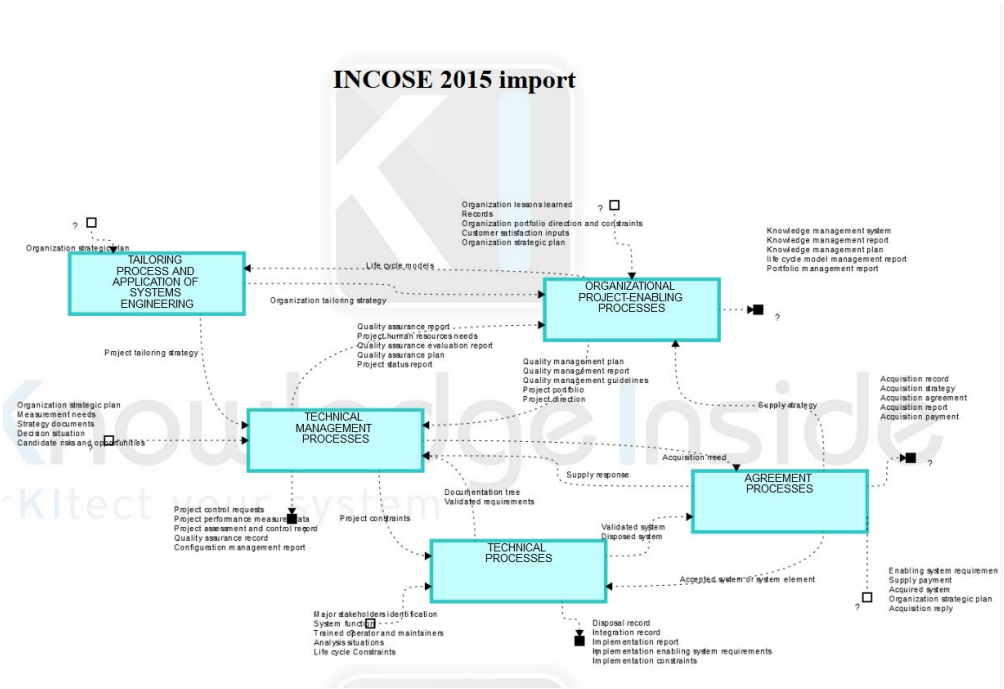
- flows between boxes are items according to ISO 15289, e.g. documents delivrables resulting from ISO 15288 processes execution.
For obvious IP reasons, we did not display definitions and descriptions from the norms.
- when double clicking on a box, you go into that box and see corresponding sub-processes.
- by convention, flows names with same producer and same destination are grouped together.
- small white boxes correspond to input ports of flows that are not coming from an actor of the same layer.
- small black boxes correspond to output ports of flows that are not going to an actor of the same layer.
- question mark on ports ("?") means that corresponding flows are not produced or consumed.
- if mark "!!" is present on a port, it means it is a double production of flow. This is possible if in the norm, same deliverable is produced by different processes. This is possible e.g. "problem report" is produced and consumed by many processes, however they do not correspond to the same kind of problems and could be named differently accordingly.
- Especially in this model flow "problem report" and only this one has been hidden because it is produced and consumed by many processes and degraded the readability of the model.

As you will see, there are many inconsistencies in the model, mainly because the syntax is the norm is much more flexible than the one you need in a model. For instance, let's search all occurrences of "requirements specification" in the model, we get this list:



As you can see, we have six flows containing requirements specification, in each case they are either produced or consummed or both by some activities. Looking at them, you will see that some of them certainly match together, however they will be considered inconsistent in the model. The point is this modeling is suggesting taking care about this in the norm itself. It's not that difficult to clean the outcomes and incomes of each activity in such a way that we know immediately for any deliverable item which process should produce and which process should consume, however at the moment, this is subject to interpretation.

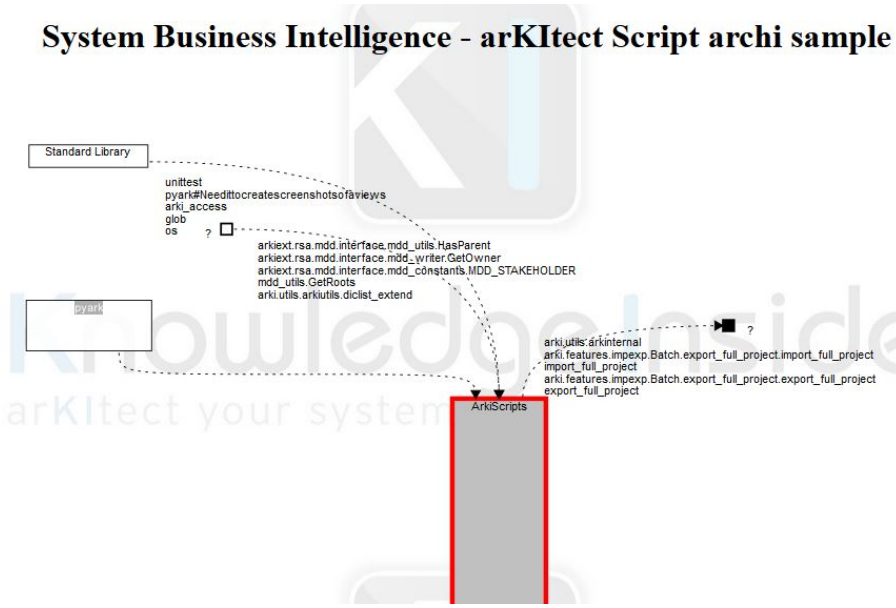
ISO 15288 as from INCOSE Handbook 2015



This web page is a model of ISO 15288 reviewed with the flows production and consumptions as reworked in the INCOSE Guideline 2015.

Conventions are same as before:
- cyan blue boxes are processes according to ISO 15288
- leaf blue boxes are activities which means parts of processes.
- flows between boxes are items according to ISO 15289, e.g. documents delivrables resulting from ISO 15288 processes execution.
For obvious IP reasons, we did not display definitions and descriptions from the norms.
- when double clicking on a box, you go into that box and see corresponding sub-processes.
- by convention, flows names with same producer and same destination are grouped together.
- small white boxes correspond to input ports of flows that are not coming from an actor of the same layer.
- small black boxes correspond to output ports of flows that are not going to an actor of the same layer.
- question mark on ports ("?") means that corresponding flows are not produced or consumed.
- if mark "!!" is present on a port, it means it is a double production of flow. This is possible if in the norm, same deliverable is produced by different processes. This is possible e.g. "problem report" is produced and consumed by many processes, however they do not correspond to the same kind of problems and could be named differently accordingly.
- Especially in this model flow "problem report" and only this one has been hidden because it is produced and consumed by many processes and degraded the readability of the model.

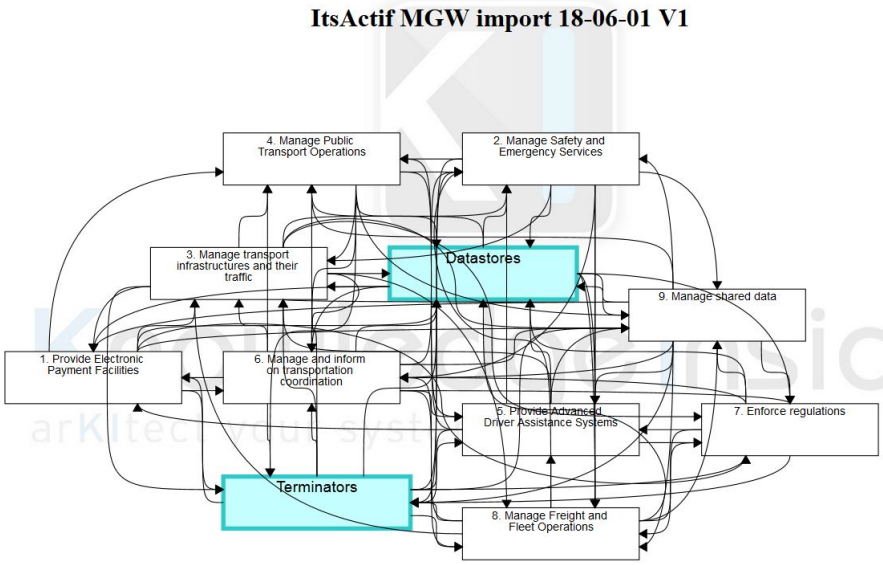
Business Intelligence on a software architecture



This view represents the hierarchical structure a source code directory (our script directory as of mid 2012).

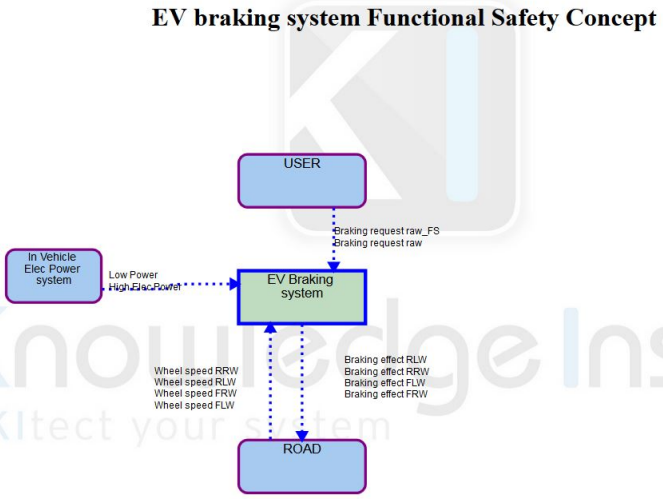
Repository boxes height are proportional to the number of line of codes behind them. Leaf objects are source files. Links correspond to "imports", e.g. use of an object defined in another file.

Import of a functional architecture on the web



This is an import of "its-actif" a web site describing all the functions needed to manage a transportation system, e.g. in a city. Each function has a web page containing its description, the input information and items and outcomes and description of sup/sub functions. Two other categories of objects are Terminators (objects or actors out of the scope of the description) and Datastores (databases needed to store data stemming out of each process activity).

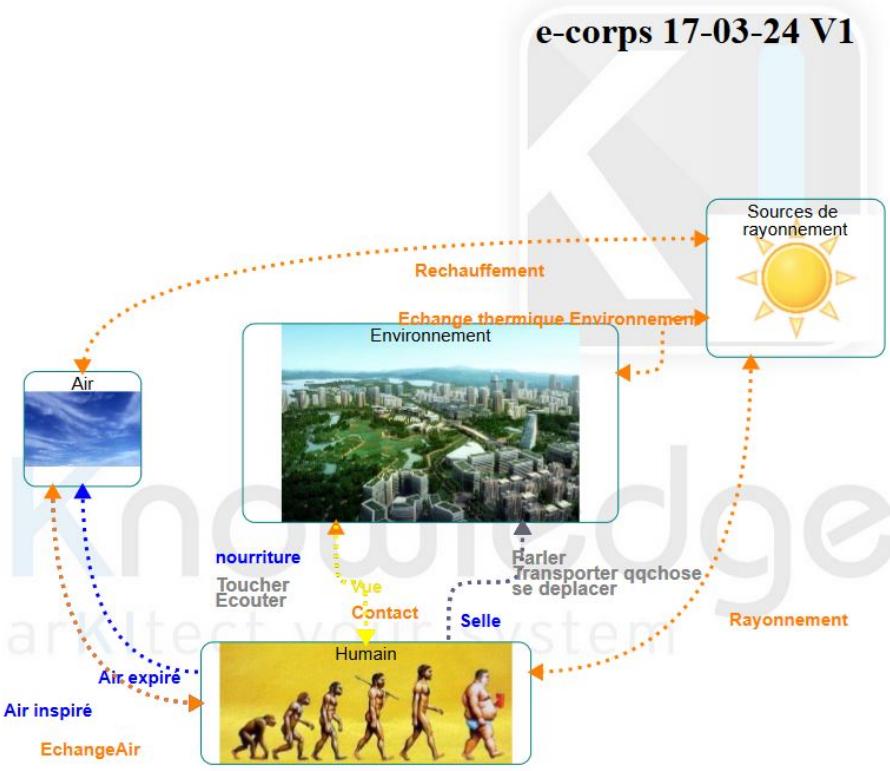
Modeling ISO 26262 Functional Safety Concept for a simplified EV braking system



In this Database, we present 5 different viewpoints: A functional architecture, A list of automatically generated safety requirements based on the Functional Safety Concept (views 2.1), a view of how the requirements are allocated to functions (view 3.2), a view of a functional chain (view 3.c) describing how the safety mechanisms are defined in order to meet a Safety Goal (No Unexpected Braking), and eventually a view 7.3 translating the functional chain into a Fault tree from which independance and safety requirements are generated automatically.

On the way to a purely digital and provable safety case compliant with ISO 26262.

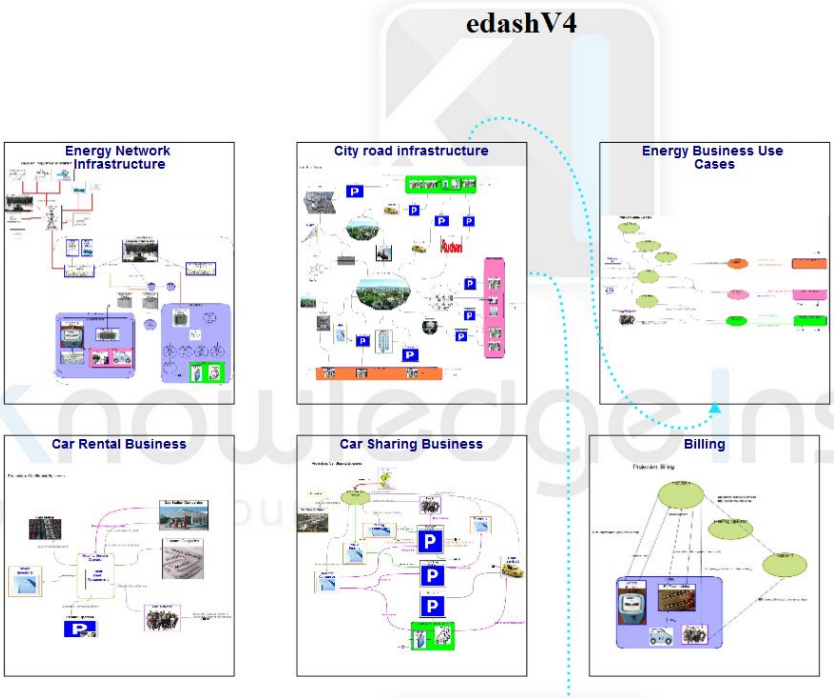
Visualizing human body hierarchically



This is a hierarchical decomposition of Human Body (In French). Double Click on "Humain" to browse into the model.

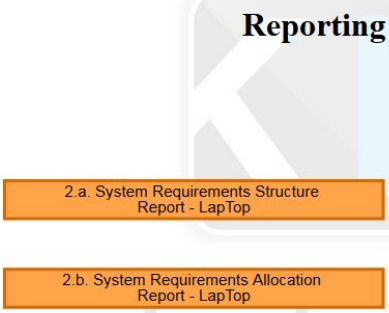
Une représentation hiérarchique du corps humain en systèmes, organes, tissus. "Double Clicker" sur Humain pour "voir le niveau sous-jacent".

Modeling Smart cities - european eDash project



Smart city, smart grid, mobility are a typical example of a complex system with various views of objects all interacting together:

Reporting



This is an extension of the Laptop model where we model reporting on the Laptop systems engineering data. Here are few samples of indicators. For each indicator, related systems engineering data is modeled. The advantage of such an approach is that we avoid a document then modification loop, e.g. direct access to requirements without description and similar nomalies allows correcting wrong attributes immediately.

List of key (and often unique) features of the platform

- fast data model definition and test (30 times faster than any modeler you may know!)
- fast creation of generative visualizations - all visualizations are maintained consistent all the time
- possibility to edit objects in visualizations
- fast interface to Excel or tabular/matrix format enabling fast import of legacy data
- extensive python interface
- user administration

- support for options and variants
- support for revision management at object and database level
- collaborative undo /redo (corresponding to a complex callback)
- last but not least, you don't need to be a programmer to use it! This may save a lot of time explaining your needs to others!