**3DSpace**

3DSpace is an Information Management System which is open and scalable which is able to support the largest, most complex, product lifecycle management deployments. It provides the flexibility to easily configure business processes, user interfaces, and infrastructure options to ensure that they meet one’s organization’s needs. The 3DSpace system operates in virtually any configuration to support your unique operating, organizational, and performance needs—on a single computer, in a networked system environment, over the internet or an enterprise intranet.

**MQL (Matrix Query Language)**

MQL is the command line interface tool for executing commands and scripts. It is similar to SQL. MQL consists of a set of commands that help the administrator set up, build, and test a 3DSpace database quickly and efficiently. MQL is an interface allows administrators to interact with enovia database. MQL is similar to SQL for enovia database. We can use MQL commands to create, modify and extract the data from enovia database.

MQL acts an interpreter for 3DSpace and can be used in one of three modes:

1. Interactive Mode - which means executing each command from the command line. This mode is typically used when you have only a few modifications to make or tests to perform.
2. Script Mode - which means using scripts to run commands. This lets you combine commands and also have a repeatable history of commands. You should use MQL scripts as long as you are in the building process. Later, you may decide to add information and files into 3DSpace. When adding information you may require additional MQL commands. Rather than entering them interactively, you can create a new script to handle the new modifications.
3. Tool Command Language (Tcl) Mode - which means you can use the tcl/tk scripting language. With Tcl embedded in MQL, common programming features such as variables, flow control, condition testing, and procedures are available.

**Frame Work:** WebUX UI

WebUX UI Infrastructure is a framework providing official Dassault Systèmes User Interface Components. These components can be classified into four categories:

* **Basic User Interface** components such as Buttons, Text Editors, and Spin boxes … Design complete applications compliant with Dassault Systèmes UI specifications.
* **Complex User Interface** components such as the DataGridView, Responsive Tile Collection View, eGraph… Represent complex and structured data by focusing mainly on the model.
* **Automatic UI Generation**: Describe the data and the best representation will be used to display them.
* **Windows Infrastructure**: Provide a flexible and customizable layout system for an application to display their data into dialogs and panels.

WebUX UI infrastructure is divided into four frameworks:

* **WebFoundation**: includes all the model parts of our infrastructure.

For example, WebFoundation contains the TreeNodeModel class that can be used to represent a complex model.

* **WebUX**: contains all the classes to represent the view of our components.
* **WebUXDataGridView**: contains all the classes concerning the DataGridView
* **WebUXEx**: which contains some components based on the DataGridView: SpecTreeDGV, AutoComplete…

**Frame Work:** UWA (Universal Web App)is a simple and elegant web apps framework that uses XHTML for structure, CSS for styling and JavaScript/Ajax for behavioral/DOM control. UWA lets developers "build once and run everywhere". Apps built with Netvibes UWA can run virtually on all major Web, computer and smartphone platforms.

**Advantages:**

* Quick & simple. Get started in minutes!
* Build once, run everywhere. UWA Apps are compatible with all major platforms (web, desktop and mobile)
* 100% compliant. Open web standards, use XHTML/XML, JavaScript/Ajax, and CSS.
* Elegant. CSS templates and JavaScript UI controls that quickly generate beautiful apps.
* Adaptive. Netvibes platform supports a wide range of devices.

**Library:** BackboneJS

Backbone is an excellent JavaScript library that you can use to write web applications in a neat, efficient manner which is based on Model-view-controller pattern. Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

Backbone's only hard dependency is Underscore.js (>= 1.8.3). For RESTful persistence and DOM manipulation with Backbone. View, include jQuery (>= 1.11.0).

Backbone consists of mainly six Components:

* Model: Models are the heart of any JavaScript application, containing the interactive data as well as a large part of the logic surrounding it: conversions, validations, computed properties, and access control.
* Views: Listens for changes and renders UI. Handles user input and interactivity and Sends captured input to the model.
* Collections: A Collection helps you deal with a group of related models, handling the loading and saving of new models to the server and providing helper functions for performing aggregations or computations against a list of models.
* Events: Events is a module that can be mixed in to any object, giving the object the ability to bind and trigger custom named events. Events do not have to be declared before they are bound, and may take passed arguments.
* Routers: Backbone Router provides methods for routing client-side pages, and connecting them to actions and events.
* Sync: Maps Backbone data to the server-side.