Design Specification SketchValid8 dekstop tester v 02

UPDATED 2024 02 04 by Jørn Watvedt, for app name: **SketchValid8**

**1: Background**

UPDATE for v 02:

Cadify must expedite and change priorities to complete this development in time for the TESS EXPO in Oslo early March 2024. Scope and task distribution has been completely changed.

ORIGINAL TEXT v 01

We are developing a new concept to improve our web response time from 1- 2 seconds to 100 – 200 milliseconds and prioritize speed and responsiveness in absolutely every line of code, and in every design decision.

Due to the development complexity here, we must first validate all our concepts and file format / syntax in a desktop environment before we commit to a full web development, hence the app name: **SketchValid8**

**2: SketchValid8 introduction**

The SketchValid8 app is a tester app that validate that HTML, JSON and CSS files are correctly assembled (concatenated) by our VBA macros in Excel, so that we are certain they display correctly in the web client later.

The CadifySketcher is a common name for a collection of UDF VBA Custom Functions and macros we have developed to run in Excel. In the bigger picture, this is a sub-system in our Cadify 3D online parametric design configurator, where the purpose of the CadifySketcher is to create instant interactivity with simplified sketches which are aligned with a 3D model in Cadify.

The purpose is to enable a mechanical engineer to use a familiar tool, namely Excel, to create these interactive sketches.

We have created several such Excel VBA Custom Function and macros as drawing tools: Lines, holes, rectangles, dimensions and so on.

Especially important is the Custom Function “CadifySheet” which define the sketch area in a millimeter-based engineering-friendly coordinate system with origin in the lower left corner.

The user can freely define the excel range for the CadifySheet, and freely adjust columns width and row height when setting up the CadifySheet.

Our macros create all the files necessary for web presentation: HTML, JSON and CSS, which we want to preview directly in the desktop tester application, **SketchValid8**, in JS with the Electron framework.

We believe that we will be able to re-use most of code and libraries in the final web application after thorough testing in the SketchValid8.

**3: Main parts of the CadifySketcher engineering UI**

Ref demo file SketchValid8 demo file 2023 12 01.xlsm.   
NOTE: This is an engineering UI ONLY, serving as the scaffolding to create the web presentation layers. Consequently, our units are mm, and origin is lower left corner.

3.1: A simulated User Interface that abstract the geometry into an easy to understand format for input parameters ( white cells are for input or selections)¨

Et bilde som inneholder tekst, skjermbilde, diagram, line

Automatisk generert beskrivelse

3.2: The CadifySheet as the actual “whiteboard” or drawing area, which holds the output that we want to instantly update in the web environment:

Et bilde som inneholder tekst, skjermbilde, line, diagram

Automatisk generert beskrivelse

3.3: The CustomFunction CadifySheet has a “settings” or input section for defining range, sheet name and so on, where the supporting “Refresh CadifySheet” button is actually a macro:

Et bilde som inneholder tekst, skjermbilde, gul, line

Automatisk generert beskrivelse

3.4: Drawing elements created from Excel “shapes”

Each drawing element, line, circle, rectangle, and so on, has a UI organized like this with a descriptive header and one element each row for simple copy-paste:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

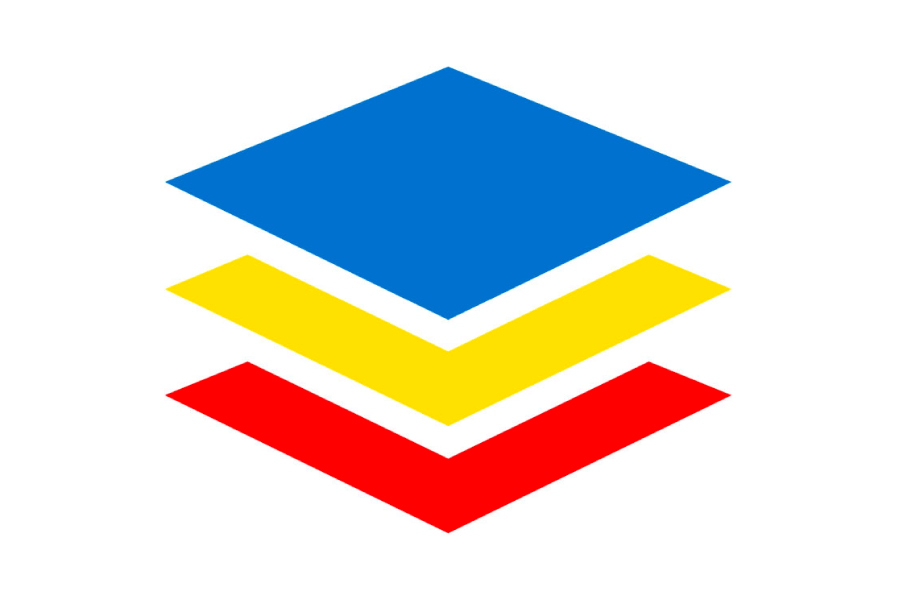
A closer look reveals a classic Excel VBA Custom Function with input parameters, where the output is a drawing element in the CadifySheet range:

Et bilde som inneholder tekst, skjermbilde, nummer, programvare

Automatisk generert beskrivelse

**4. UPDATED V 02: Description of the 3 layers of CadifySketcher content**

The CadifySheet consist of 3 distinct separate layers that has its own pipeline from within the spreadsheet on the web server all the way to the screen presentation on the web client:

Layer 1 = The SVG drawing layer as html  
Has all svg elements including svg image and svg text in a html file, with references to external images. This layer also defines viewbox width x height in mm

Layer 2 = The JSON layer = dynamic content  
Contain grid cell values driven by formulas and any conditional formatting styling as inline styling

Layer 3 = The CSS layer = static content  
All cell styling AND column width and height and all cell content for static values ( not formula driven)

File name for the HTML – JSON – CSS files are identical and retrieved directly from the Named Sheet cell in CadifySheet function: In this example, sheet name is “Page 1”:

Et bilde som inneholder tekst, skjermbilde, line, nummer

Automatisk generert beskrivelse

Consequently, the output files are named:

Page 1.html

Page 1.json

Page 1.css

There may be an unlimited number of instances of CadifySheet in a workbook, but each instance must have different “Named Sheet” names.

4.1: Layer 1 = The SVG drawing layer as html

Every excel shape is converted into a svg element html in accordance with the svg element html specifications:

<https://www.geeksforgeeks.org/svg-element-complete-reference/>

<https://www.w3.org/TR/SVGTiny12/intro.html>

( possible future option if we go all js: <https://svgjs.dev/docs/2.7/> )

Example of svg html code generated by our Excel vba macros:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Grid line styles are defined in the CSS

IMPORTANT: Image handling is not yet fully defined, but all images are identified in Excel, saved to disk, and a list of images is created as a parameter in CadifyImage. We are considering using Dropbox, Cadify Library or NopC Postgres DB.

4.2: Layer 2 = The JSON layer = DYNAMIC LAYER

In this context, the definition of “dynamic” is any styling and data that may change during run time in the client/server refresh context:

* Cells having excel formulas
* Conditional formatting

The JSON format for grid content is not yet fully defined, and shall consider the following issues:

* The JSON file is created programmatically by VBA / C# in Excel
* Our grid content is generally sparse and unstructured compared to more classic excel tables.
* We may or may not later decide to create a process for dynamically loading JSON file with changes only, after initially loading the full JSON file.

For our application, the content is generally sparse, with a few cells having actual content, so our JSON format is created so that each cell is addressed individually in Excel A1 format, starting in upper left cell. Then only those cells that has actual content get exported by our VBA macro into the JSON file in this format:

* Freely follow the format of content in the “Excel Export Range As HTML”
* Each cell is ID’d and runs from upper left to bottom right, row-by-row, starting with “A1” in upper left cell.
* IF cell has no formula and there are consecutive identical cells, or consecutive empty cell, ONLY first cell is specified in the JSON, then we assume cells between up to the next reference, are identical.
* Dynamic Inline CSS = Conditional formatting as inline CSS to be added in Rev 04 mid January: Conditionally formatted cells overwrite whatever “static CSS” styling has been set in the CSS.

Example with columns and row ID’s in red boxes:

Et bilde som inneholder tekst, diagram, Font, line

Automatisk generert beskrivelse

Rough JSON syntax mockup:

"cells": [  
 {

“ref”: “B9” ,

"content": "ø30"

},

{

“ref”: “C9” ,

"content": ""

}

{

“ref”: “E9” ,

"content": "27"

}

{

“ref”: “F9” ,

"content": "ø 30"

}

]

4.3: Layer 3 = The CSS layer = STATIC LAYER

In this context, the definition of “static” is any styling and data that is fully defined during design time and never change during run time in the client/server refresh context:

Grid column width and row height in mm: The grid gross size shall fit exactly in the Viewbox as defined in the HTML section ( scaled to fit)

Merged cells

ALL default cell text and styling

The grid itself is a (scaled to viewbox) reproduction of the Excel range, which we want to define in a minimalistic js library: <https://gridjs.io/>

Et bilde som inneholder tekst, Font, nummer, programvare

Automatisk generert beskrivelse

Our “Cadify CSS” shall be “cascading” in the sense that the first = upper left cell, shall have complete definition of all relevant properties, which on the client side is then set for the entire CadifySheet range. Then the subsequent CSS sections overwrite or supplement this initial first CSS section for any subsequent changes.

* Freely follow the format of CSS in the “Excel Export Range As HTML”
* Column width and row height are considered “static” and shall be included in the CSS.
* Cell content for non-formula cells are considered “static” and shall be included in the CSS
* CadifySketcher shall be all in mm, and CSS must use width and height in mm as defined in the CadifySheet as integer mm.
* Each cell is ID is referenced iaw Para 8, and runs from upper left to bottom right, row-by-row.
* CSS section for Upper Left Cell shall contain defaults where defaults must be defined.
* Redundant CSS information is omitted.
* For Top Row there must be one CSS Section for each column width
* For Left Column there must be one CSS Section for each row height
* Merged cells shall be managed in the CSS. Example for 2 merged cells: colspan="2">Merged Cells</td>
* Other than first row and first column, where cell width (column width) and cell height (row height) is defined, it should not be necessary to include in other CSS sections.
* Static cell content shall be included in the CSS: IF cell HAS NOT “=” sign AND value is NOT EQUAL “blank” then value must be included in the CSS.
* Do not worry about conditional formatting, which will be managed by inline CSS in the JSON, and then overwrite whatever static styling has been defined in the CSS.
* IF CSS B1 = CSS A1 then skip, moving in a cascading sequence from upper left to lower right cell.
* IF “CSS nn” has any different styles than “CSS nn-1” then ONLY THOSE DIFFERENT styles shall be included in the “CSS nn”
* Suggest that last cell in the CadifySheet Range shall have a CSS sections that state this as the last section. ( no need to loop thru any further validation)

**5: What is the purpose of CadifySketcher?**

Create online instantly updated sketches / drawings for our products. In this illustration the product is a simple base plate with a hole pattern. On right hand side there is a column of controls to modify length / height / hole diameters / patterns and so on:

Et bilde som inneholder tekst, skjermbilde, diagram, nummer

Automatisk generert beskrivelse

**6: How to play with Cadify Sketcher in the demo file**

Ref. demo file: SketchValid8 DEMO 2024 01 04.xlsm

The orange color rows at row 13 and 14 are simply a header for the various input paramters to the UDF “CadifySheet” which define the sheet range and works together with the macro “Refresh CadifySHeet” to calculate Excel column width and row height from Excel units to mm.

For the attached sample, we have reduced the actual CadifySheet “Sheet Range” to just four cells = range D6:E7 with content “Hello world”

Manually change the range in “Sheet range”, press the “Refresh CadifySheet” button and see how range changes. Even more visible if “Sheet guidelines” are turned on.

MOST IMPORTANT: Output Max X and Max Y calculate the actual column width and row height from Excel values to mm:

Et bilde som inneholder tekst, line, nummer, Plottdiagram

Automatisk generert beskrivelse

Lines are managed by the UDF “CadifyLine”, as shown here for the 4 lines surrounding the current D6:E7 range. Line style allow different styles, and use “suppress” to turn off:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

**7: SketchValid8 development stage 1: App and initial form with UI**

SketchValid8 shall always co-exist with our CadifySketcher, which is embedded in Excel, because the purpose is simply to validate the HTML + JSON + CSS files from Excel

Et bilde som inneholder tekst, håndskrift, Barnekunst, blekk

Automatisk generert beskrivelse

Buttons “HTML”, “JSON “ and “CSS” save these files to the same folder as that excel itself resides in = “WB folder”, where SketchValid8 pick them up, display them in the Electron JS APP form.

NOTE: Currently, only the “HTML” button has been developed, while the “CSS” button is expected soon.

Here is a mockup of SketchValid8, for the template CadifySheet, where width is 42 mm and height is 11 mm, shown to approximately correct scale, which represent the source data for html viewbox: <html><head> 13:09:34</head><body> <hr /> <svg width='42mm' height='11mm'>

Et bilde som inneholder tekst, Font, skjermbilde, nummer

Automatisk generert beskrivelse

File



For the current DEMO there are two pages: Page 1 and Page 2, which produce two different sets of output files, as shown here: ( HTML only for now, JSON and CSS later)

Et bilde som inneholder tekst, skjermbilde, Font, line

Automatisk generert beskrivelse

Required functionality and deliverables for Development Stage 1:

* Installer file for SketchValid8 on Windows PC, including all required files all wrapped into a single installer.
* Build a form as shown above, that is stretchable width and height
* Button “File” that browse to a folder that hold the HTML / JSON / CSS files, and select one Sheet Name to be displayed ( In our example: Either Page 1 or Page 2)
* Buttons check buttons to select HTML file for display ( Later this is more relevant when we have the complete set of HTML / JSON / CSS and the possibility to turn each one of them on/off and put them on top of each other)
* Read the HTML file created by CadifySketcher, create a viewbox sized in mm and add the svg element lines to same scale.
* Viewbox with a gray background to be displayed to scale in center of the form for: <html><head> 13:09:34</head><body> <hr /> <svg width='42mm' height='11mm'>
* Viewbox shall scale DOWN, while keeping aspect ratio, if form width or height is mouse stretched so that scale 1:1 won’t fit anymore.

**8: SketchValid8 development stage 2: Add grid.js and static styling to cells in grid from CSS file.**

Here is how to export a native Excel html file for our range D6:E7

|  |  |
| --- | --- |
| Select range |  |
| File / Save As / Selected range |  |
| Result is a htm file that you open in Notepad etc: |  |
|  |  |
|  |  |

Here is copy of CSS section of that file for cell D6:

.xl1528560

{padding:0px;

mso-ignore:padding;

color:black;

font-size:11.0pt;

font-weight:400;

font-style:normal;

text-decoration:none;

font-family:Calibri, sans-serif;

mso-font-charset:0;

mso-number-format:General;

text-align:general;

vertical-align:bottom;

mso-background-source:auto;

mso-pattern:auto;

white-space:nowrap;}

Here is a mock up of how we expect to create our CadifySketcher CSS file, shown for cell D6, with our special syntax in yellow highlight:

#CellD6 {

width: 21mm; /\* Set the width of the cell in mm \*/

height: 5mm; /\* Set the height of the cell in mm\*/

padding: 0px;

mso-ignore: padding;

color: black;

font-size: 11.0pt;

font-weight: 400;

font-style: normal;

text-decoration: none;

font-family: Calibri, sans-serif;

mso-font-charset: 0;

mso-number-format: General;

text-align: general;

vertical-align: bottom;

mso-background-source: auto;

mso-pattern: auto;

white-space: nowrap;

position: relative;

content: "Hello";

}

Our draft design rules for the CSS syntax in acc with Para 4.3

Scope and deliverables for SketchValid8 development stage 2:

* Work with Jørn at Cadify to develop the CSS syntax / file format, including for merged cells.
* Use the js library gridjs.io to build a grid as defined by our CSS files: <https://gridjs.io/> (Custom Grid instead)
* Find best possible way to specify cell width / height in gridsjs in mm. Suggest checking out our CadifySketcher macros on how to convert in excel to mm.
* When loading the HTML file, as described in Stage 1, now also simultaneously load the CSS file, and indicate in the checkbox button ( color green ?) that the CSS file is loaded, and when checked then load in the viewbox to exactly fit in the viewbox.
* Fine tune scaling so that viewbox mm for SVG elements match mm for grid.
* Updated installer file
* Source code as VS Project zip file with inline comments in English.