SPREADSHEET MODELLING & DOCUMENTATION

A note about how to build valid & robust models with spreadsheets

Summary:
This note describes how to build solid and valid spreadsheet models that are well documented; validated and can be used by other users than the person who created the model. So using the recommendations in this note will be a good start making sure that a model can evolve and be maintained by multiple persons and users over time.

This note is not about how to build models in a specific program as the suggestions are generic but from time to time references are made to e.g. using Excel.

Version 3.1

Table of content

Modelling the real world

Representation of the system and model concepts

A model in the following is defined as a simplified representation of the 'real world'. The model has some "borders" that sets the limits in relation to the system represented against all the rest of the real world – meaning what is 'inside' the model and what is outside the model.

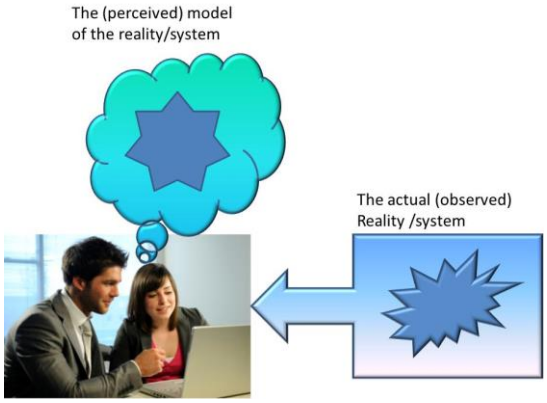


Figure 1.

System concepts

A model or a system can in generalized form have components like shown in figure 2 below:

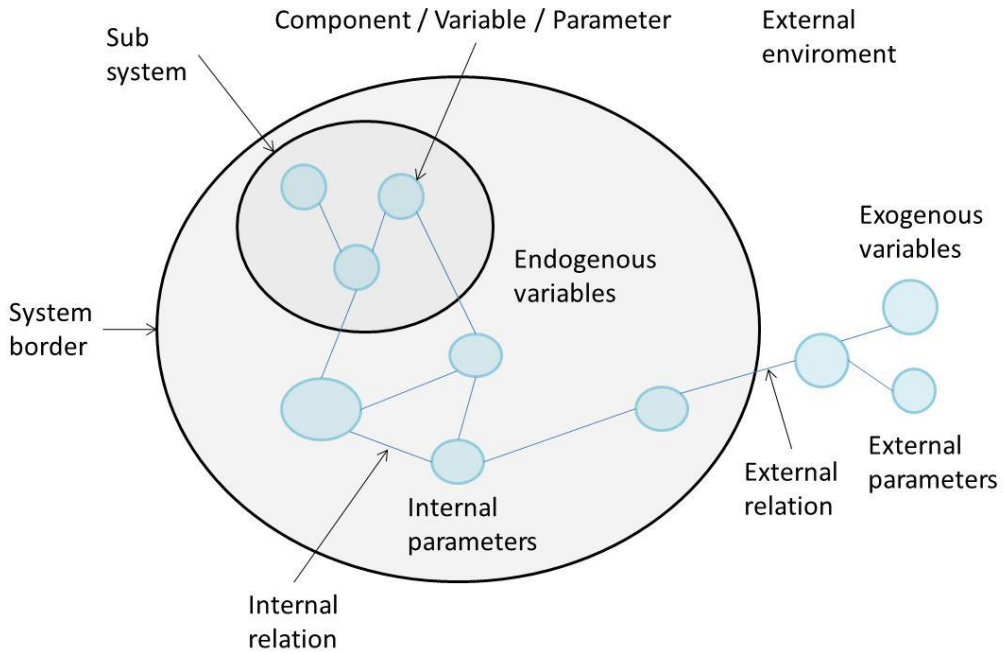


Figure 2

The model or 'system' has a system border which defines what is expected to be 'inside' the model, and what is 'outside' the model. However, the total model can operate with 'outside parameters and variables as shown above. In most cases outside parameters and variables will be limited compared to the internal variables and parameters.

A parameter is in this note defined as a holding place for a constant that has influence on the model transformation and is used in functions (e.g. f(x) where f=function and (x)=parameter that is used in the function). An internal parameter is observed as a constant within the scope of the model and therefore inside the systems/models border. An external parameter is likewise a constant that just are observed outside the system border but can have direct influence on the model (e.g. expected country inflation rate that might be an external parameter to the defined system and have influence on the models calculations).

A variable is in this note defined as a holding place for data that can be transformed by functions in the model. Input variables can be data that is imported and used by the model and the variables will by the models functions be transformed to stage variables and final output variables. Like parameters variables can be observed both as within scope of the models borders and outside the borders as external variables.

A function is a mathematical "receipt" of how some data (like input variables) will be transformed and the function will deliver or return transformed output value. Input to a function can be several input variables as well as one or several parameters while the output only can be a single value. In figure 2 the term relation is considered as representing the situation that some input variables and parameters are connected and can be transformed to output variables by functions.

Models in spreadsheets

Many models in spreadsheets have some variables and transformations over time. In many cases time is represented in columns with each "time-unit" (hour, day, week, month, year e.g.). Data or variables and parameters are then shown in each row like in figure 3 below.

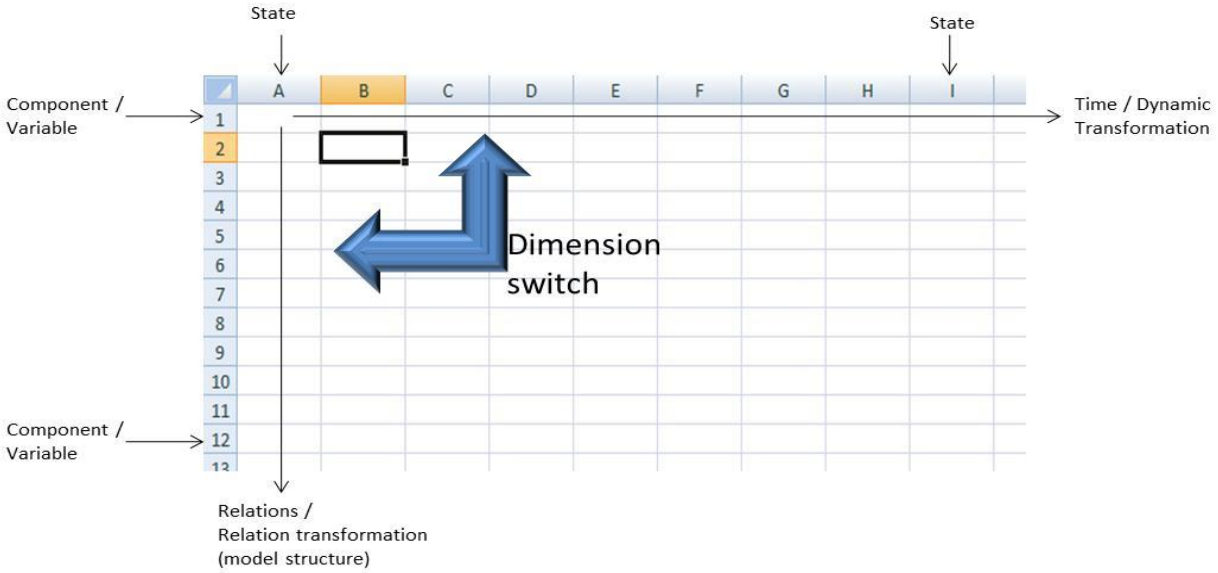


Figure 3

Of course these 2 dimensions can be switched instead depending on the model builder's preferences and what makes most sense for the model.

So in some (many) cases mathematical models in spreadsheet has both an internal variable transformation within the model logic as well as a time transformation dynamic.

Challenges with spreadsheet models

Spreadsheets have won extreme use as 'model' builders since they got popular in the early 1980ties. This is mainly because of the intuitive easy logic that a spreadsheet represents a 'normal' piece of paper. You can type in input data, you can type in functions and then the system can show you the output of the function. This concept is extremely easy to build and intuitive to use. Likewise spreadsheets have built-in many formatting features that make it very easy for a lot of users to create very nice reports and presentations of the shown data.

See figure 4 below.

However, with big models the overview of the model decreases and many spreadsheet models has been abandoned because the model had to be "maintained" by someone other than the original 'model builder'. Only the user that made the model had the overview and the knowledge and understanding of the concepts build to operate the spreadsheet and the model would be too complicated to use for others.

Another problem is when someone suddenly finds an error with the model output. If the model is really complicated and 'messy' then it can be very difficult to find where the error originated.

Overview of the mathematical model

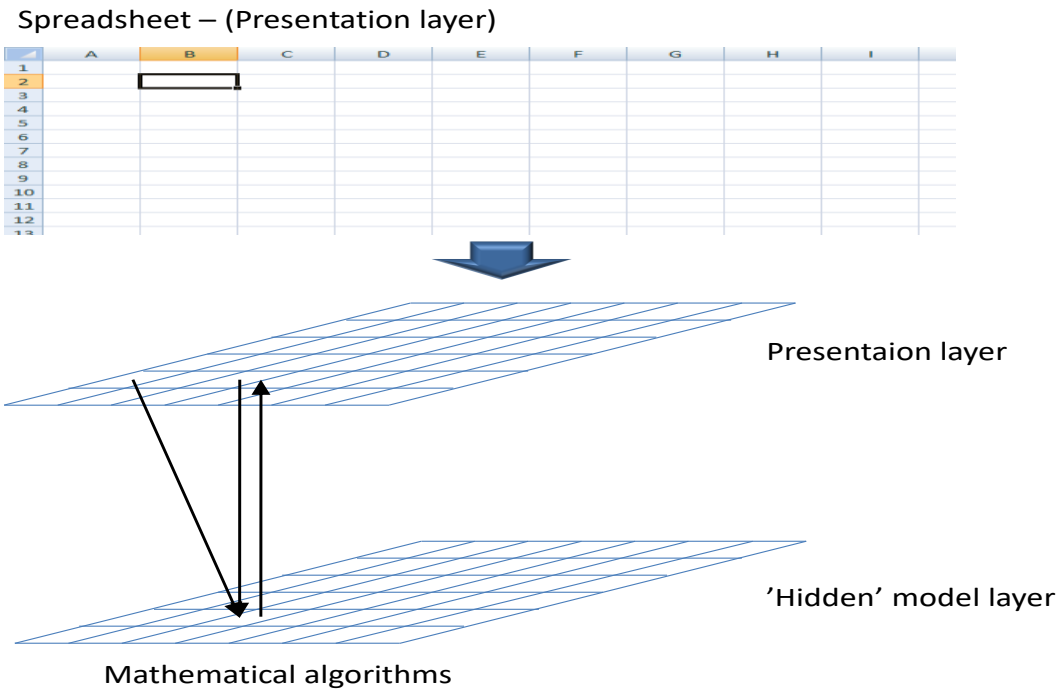


Figure 4

The concept in spreadsheet is that you can have input data and output data shown and formatted nicely and easily and therefore makes an ideal tool for making reports and presentations of data while all the functions and formulas are “hidden” from the presentation layer.

This is however also one of the biggest problems with spreadsheets. The actual mathematical model is “hidden” in each cell and not easily represented. The situation where each cell has its own formula (or function sending one output to the presentation layer) gives another problem as when many formulas are supposed to be the same through a row or a column then this is not necessarily the case if the model builder makes a mistake or by accident changes or overwrites one or more cells with different or changed formulas that was not intended.

Spreadsheet visual size

Another problem is that spreadsheets have evolved from a few hundred lines and columns to be possible millions of rows and columns and a model builder can make huge models that can’t be seen og shown by one computer screen but you have to scroll left and down to find all the data and all the model transformations making it very difficult for users not familiar with the model to get an overview of the model and how if functions. See below in figure 5.

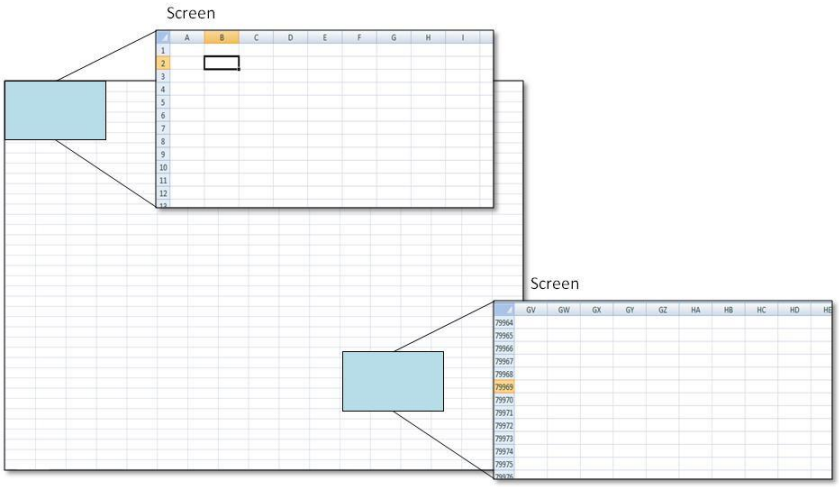


Figure 5 (huge areas in spreadsheet and a small screen)

Build and document spreadsheet models – Best practices

Overview of how to build spreadsheet models

In figure 6 below a "model" is presented of how to organize a spreadsheet model so it is systematic in its logic and hopefully much easier can be used by other than the model builder giving the possibility that the model will 'live longer' than only one user. Likewise, it also gives some ideas of how to have better validation and checks of the model transformation.

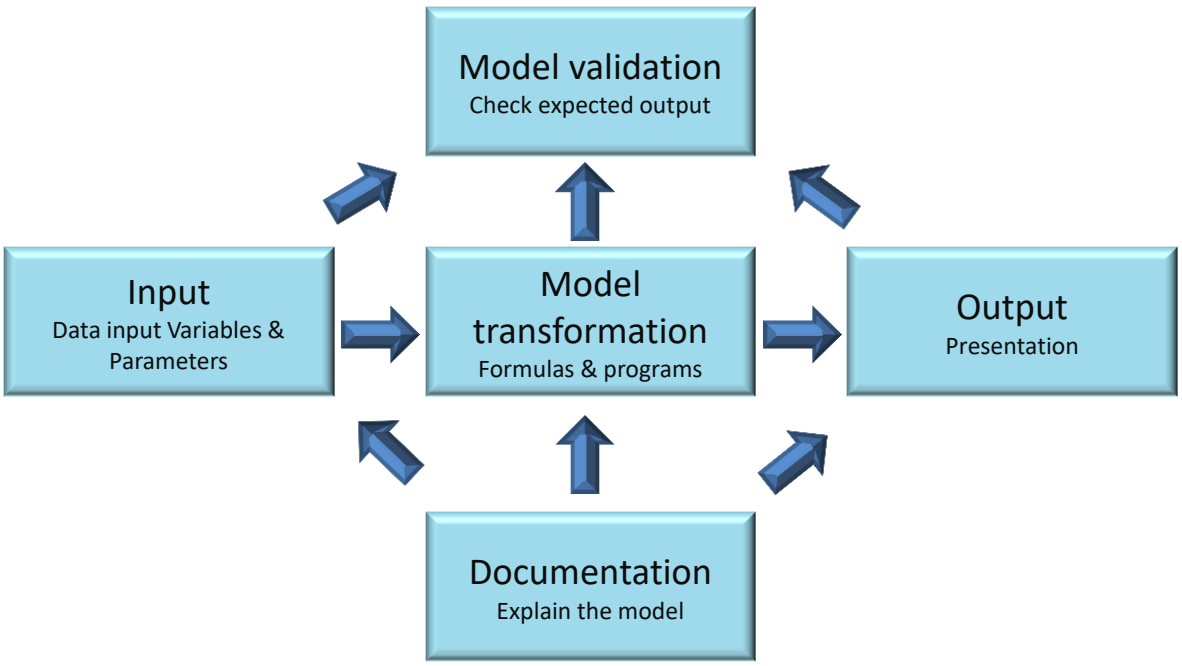


Figure 6 (General description on how to build a robust spreadsheet model)

Figure 6 introduces the concept that any spreadsheet model should be portioned into 5 major areas:

- Input variables and parameters (in the following split into 2 areas)
- Model transformation (formulas/Functions)
- Output area / presentation and print-outs
- Model validation (validate formulas and input variables)
- Documentation (make additional documentation to the model)

To be even more systematic it is suggested that variables and parameters will be divided even more so variables have one area of its own and parameters an area of its own so that the final model ends up with 6 main areas

Depending on the complexity and the number of formulas and data input and output we can have 2 approaches (and a lot of in-between solutions in how this is done in a spreadsheet.

- Single sheet approach
- Multiple sheet approach

In the single sheet approach all areas are handled within a single sheet as shown below in figure 7.

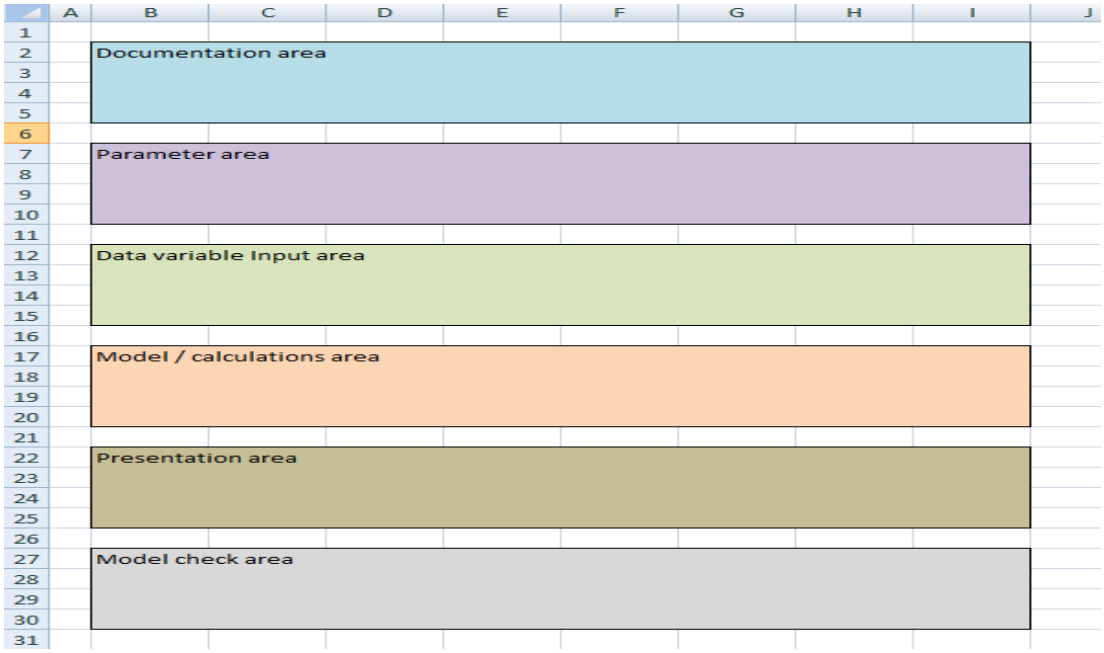


Figure 7 (Single sheet design)

This is not that common as any normal used spreadsheet today has multiple sheets and it makes sense to use this feature as shown below in figure 8.

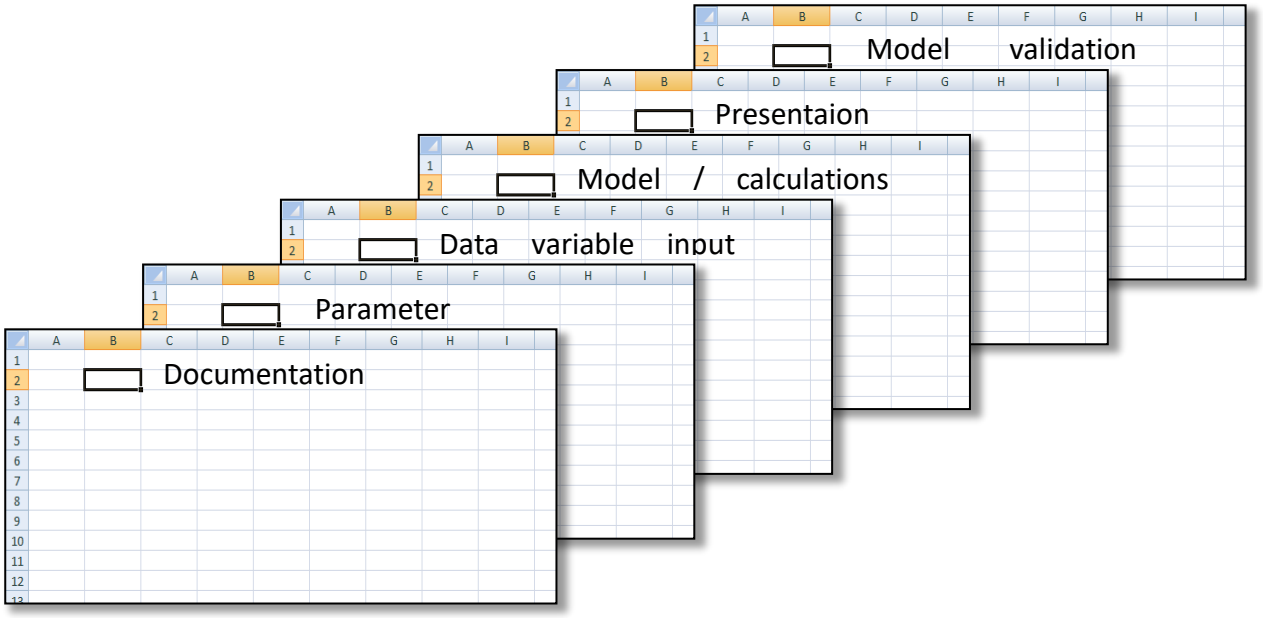


Figure 8 (Multiple sheet design)

As stated earlier then this can of course also be a mix where we place parameters and variables in the same sheet and/or consolidate some other sheets into one. In other cases where we have very complex models and we can have sub-models it might make sense to use even more sheets so e.g. variables will be in multiple sheets. In even more complex situations, it might be best to split sub-models into workbooks/files of its own and then have a master spreadsheet that gets results from the sub models that are placed in each a spreadsheet file (multiple spreadsheet-file design).

Design process

When a design of a model shall start then as in most programming and design phases always start to focus on the deliverable – in this case: what is the wanted output and presentation. When this have been defined then focus on which parameters the model is expected to use. That means which constants will be used within the program. This can be interest rates, starting year, and all other constants that need to be defined for the model to work.

After that focus on which input data that needs for the model to work. When this has been done then the actual model and transformations can be designed.

Any model be validated so there should be designed some checks to make sure that the output will be as expected. When all this has been defined the model needs some documentation so that users of the model easily and quickly can get an overview of the model.

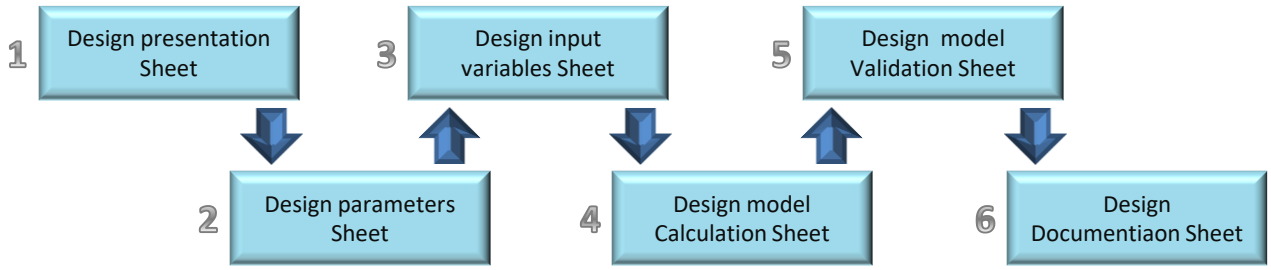
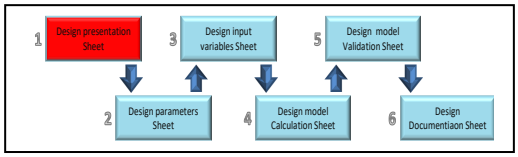


Figure 9 Design process

In the following each of these steps will be described in more detail and it is assumed that we use the multi-sheet design approach.

Step 1 – Presentation sheet



First step is to design the expected output. By doing so you will get information and typically overview of both input variables; parameters and transformations and calculations.

The recommendation is to have this sheet/area completely 'clean' of any data input or transformation. This mean that this area is only for the purpose of what it says: Presentation.
This sheet only has formatting and + (plus) formulas getting info from parameters, input, and model sheets.

An example of a very small simple model:

Revenue increase per year:	4,40%					
Expense increase per year:	3,35%					
	2009	*Index*	2010	*Index*	2011	*Index*
Revenue	11.210	100	11.703	104,4	12.218	109,0
- Variable expenses	3.435	100	3.550	103,3	3.669	106,8
= Contribution margin	7.775	100	8.153	104,9	8.549	110
Contribution margin %	*69,4*		*69,7*		*70,0*	

Table 1. (Simple model)

This very simple example shows a wanted output (including data) and if this model was not bigger it would make no sense to split this in a lot of sheets but as an example of the principles it is OK for now.

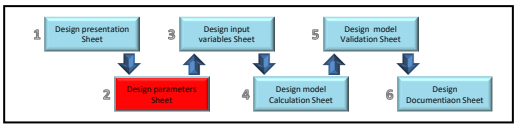
In this example it is expected that ALL the data in the table comes from underlying sheets and there are made NO calculations inside the table. It only consists of simple "+ formulas" getting the data from the other sheets. The purpose of this sheet is to present the output – not to make input or calculations. In other words – it is repor

From the design we can identify 2 input variables and 2 external parameters:

Input variables: Revenue 2009 and Variable expenses 2009.
Parameters: Revenue increase per year and Expense increase per year.

Revenue and expenses in 2010 and 2011 are calculated based on the parameters. Likewise is every other number in the table calculations that should be placed in the model sheet.

Step 2 – Parameters sheet



When the Presentation sheet has been designed then you can start by a listing of all parameters used.

This sheet should be split in at least 2 areas: Internal parameters and External parameters. See definitions of parameters earlier in "System concepts".

A parameter can consist of a single value that has influence of the whole model, or a parameter can hold many values (in some cases one value per time period). A parameter influences and is input for transformations / functions but will not be changed by functions. A parameter is a constant and a variable is as it says a holding place for something that varies.

In the example from the "Presentation sheet" we have identified 2 parameters. We can't by the Presentation alone tell if these 2 parameters are internal (some actions we do inside the scope of the business modeled) that influences the increases or it is some external influences like inflation e.g. In the example below it is assumed that both are external parameters:

PARAMETERS						
External parameters						
Yearly revenue increase %	4,40%	A combination of …..				
Yearly expense increase %	3,35%	Taken from …				

Table 2 (Parameter sheet)

Make the text of each parameter as descriptive and clear as possible. It is also a possibility to have some additional information written to each parameter so it is as clear as possible to the reader what each parameter covers.

Parameters Good Practice 1: - Parameter descriptions

It is also a good practice to make a description of where each parameter is used in the functions / transformations in the model sheet.

Parameters Good Practice 2: - Update documentation for each parameter you create

While listing each parameter you can make the documentation in the documentation sheet as you go.

Parameters Good Practice 3: - Create different presentation languages

If you should use different languages (Language translation) for the presentation area you could make a language area in the parameter area where you have each language in e.g. a column for each language and then you could have another parameter where you choose the language and in the presentation area you make a "lookup" based on the language parameter choice and find the needed language for each piece of text in the presentation area.
In this way you do not need to translate e.g. a balance sheet from one language to another manually but can just have formulas replacing the language based on parameter choices. As each piece of text will be fixed information you either place the different languages in the parameter area or make a language info sheet of its own.

Parameter Good Practice 4: - Make starting point (year, month, week e.g.) dynamic

If the model will/shall be dynamic in the sense that years or month will change over time then it is a good practice to make the whole model dynamic and that the time starting point will be specified by a parameter like in the enhanced example below:

PARAMETERS

External parameters

Yearly revenue increase %	4,40%	A combination of …..
Yearly expense increase %	3,35%	Taken from …

Internal parameters

Starting year	2010	Type in the start year of the model

Presentation language	Danish	Choose either "English" or "Danish" from the list

Translations	English	Danish
1	Revenue inc	Omsætingsvækst per år:
2	Expense inc	Omkostnings stigning per år:
3	Revenue	Omsætning
4	- Variable e	- Variable omkostninger
5	= Contributi	= Dækningsbidrag
6	Contributior	Dækningsgrad
7	Index	Indeks

Table 3. (Extended parameter sheet with 2 languages)

By using a "starting year" the whole model with both model sheet, input variable sheet and presentation sheet will depend of the starting year. The presentation would then instead look like this and please note that presentation language in the parameter section has been changed to Danish so now the model text has been converted to Danish in the presentation area:

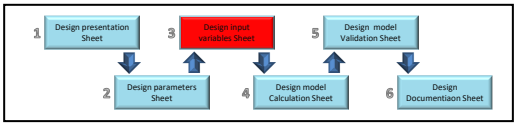
Omsætingsvækst per år:	4,40%
Omkostnings stigning per år:	3,35%

	2010	*Indeks*	2011	*Indeks*	2012	*Indeks*
Omsætning	11.210	100	11.703	104,4	12.218	109,0
- Variable omkostninger	3.435	100	3.550	103,3	3.669	106,8
= Dækningsbidrag	7.775	100	8.153	104,9	8.549	110
Dækningsgrad	*69,4*		*69,7*		*70,0*	

Table 4. (Presentation sheet with Danish as language)

Note: You would not let the user type in the language choice but would make a drop-down box with the available languages to choose from made out of a list.

Step 3 – Input variables



As with the parameters we can have external variables (also called exogenous variables) and internal variables (endogenous variables).

Input variables is typically some input data but information where each variable can and will have different values which will be changed by functions / transformations done in the model sheet. As a difference to parameters which will not be changed by the model then variables can hold different values. This can be different values over time but also different values depending on other factors than time (like different parameters; different input variables or different scenarios by simulations – see more about this later).

Again if we look at the first example we have 2 input variables holding Revenue in 2009 and Variable expenses in 2009. Both revenue and expenses will change in 2010 and 2011 as a result of the parameter values.

INPUT	
	2009
Revenue	11210
Variable expenses	3435

Table 4. (Input sheet)

Variables Good Practice 1: - Descriptive input parameters and comments when possible.

As with the parameters it is good practice to make the text of each variable as descriptive as possible and likewise with documentation of where each input variable is used in the model sheet and of course update the documentation sheet as you list every input variable.

External data import (databases e.g.)

In some models the input data can be typed in and in these cases it is simple manual input but in more complex models you have a lot of input data and in some cases these data comes from different sources.

It is often possible to import data from many different sources like:

- Other spreadsheet files
- Databases (like Access, Oracle, MS SQL server e.g.)
- Internet connections
- Text files (.txt; .csv and others)

e.g.

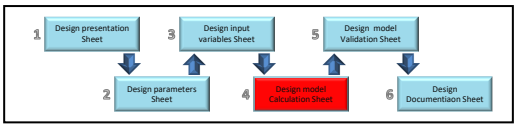
Variables Good Practice 2: - Import data if possible to avoid manual input (avoiding errors)

As a general rule you should import data from the source if possible instead of making reports and typing in the data manually. Obviously because of 2 reasons: It will normally be much quicker to import directly than typing in manually saving time and what is even more important you will remove the possibility of having typing errors of the input data.

Variables Good Practice 3: - Automate import when possible

In some case's the tasks of importing, is the same every time the data will have to be imported.
In these case's it is recommended that the procedure is automated as well. Again, to save time but also to make sure that the import process will be the same every time.
Automation of these processes can be made by either recording a macro in the spreadsheet or by using one of the many free scripting/automation languages that can automate e.g. Excel using the COM model.
If you record your import process it is a very good practice to make comments of the program afterwards describing each step of the import process so this can be change by others in the future creating a new import process if needed.

Step 4 - Model & calculations



The "hart" of a model is of course all the calculations; transformations and functions done to create the output for the Presentation sheet. Most model errors are of course also happening because of errors in the formulas of these transformations.

Formulas

The formulas transforming input variables and parameters to some output variables is where we have the most errors in models. In the following some recommendations will be described so the model will be easier to build, maintain, validate and document.

Formula Good Practice 1: (One formula per row/column).

Have only one formula per row / coloumn.

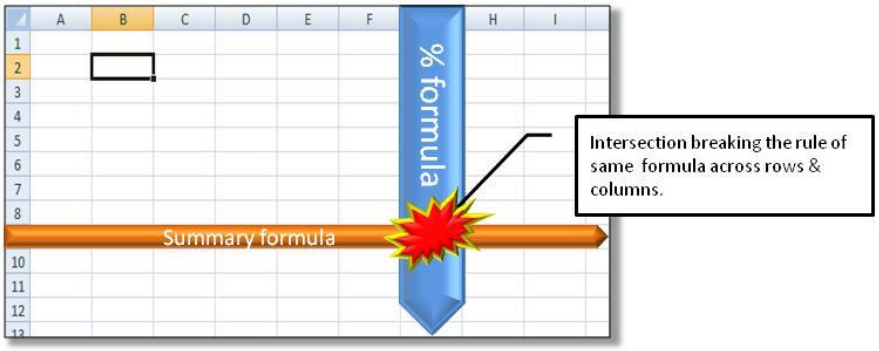


Figure 10

The reason for this recommendation is that if you have different formulas across of a row or a column then the possibility of an error increases many times while both building but also later maintenance / changes will be made while copying formulas.
It is much easier to make a formula once – check its validity - and then copy it across a row or a column entirely.

In the original "presentation example" we had some figures with input variable per year but also with an index per year. In this example we have different formulas in the columns as shown in the extract below:

	2010	*Index*	2011	*Index*	2012	*Index*
Revenue	11.210	100	11.703	104,4	12.218	109,0

Table 5. (example of mixed data types in the same row)

Building the model the revenue for each year should be calculated in one row and the indexes should be calculated in another row.

Formula Good Practice 2: (Group formulas and then have formulas in 'natural order')

While building the model then make the calculations in a 'natural order' from left to right and from top to bottom compared to the expected presentation and the expected 'order' of natural calculations.

However, as we look at the example and have to follow the abovementioned rule of only one formula per row/column then you can't have everything in the same order as the presentation area. We need to follow one formula per row/column first and then take the calculations in the natural order when possible.

Likewise, as it will be clear later then we should first group the different types of calculations so we have common types of calculations together but inside each group we should follow the rule of 'natural order'.

Formula Good Practice 3: (Simple formulas – split complex calculations – Transparency)

In many cases there has been made complex models with extremely complex formulas. It can be formulas with 3-4 (and many times many more) nested IF, THEN, ELSE conditions with the same number of ending parenthesis and in worse cases with additional formulas inside like horizontal or vertical LOOKUP's in input data. In these case's we end up having many problems. It is very difficult to decipher the transformation and because of this it is of course also a possibility for errors. But even worse as the formula is very difficult to 'understand' maintenance will be difficult.

In many cases the reason for having complex formulas is because you mix the model area with the presentation area. You try to build all the transformations directly in the presentation area and then of course you can't have intermediate calculations (unless you use hidden rows or columns which is not recommended – see FGP 9). While separating the presentation area from the model area it makes much more sense to make calculations transparent and make intermediate results and have the calculation broken into more columns or rows. In some case's you could ending up having e.g. 5-15 rows (or columns) of calculations (instead of having it all in one complex formula) with a description of each row of calculation and having the last row as the end result. By doing so it is much easier to:

- Document the transformation of input data / parameters
- Decipher the formula and understand what happens.
- Validate the transformation and output
- Maintain the model in the future for another model builder

As a general rule, use only one function per row/column (do not nest several functions per row/column). It will be much easier to understand each sub-calculation and much easier to maintain.

Formula Good Practice 4: (Document each formula)

Document each formula. Make documented code.

Make the documentation of each formula directly with each transformation. As described in FGP3 then each line or column of calculations should be documented telling what the formula does. If the calculation is rather complex you can make a reference to the documentation sheet where you make a more comprehensive description.

Formula Good Practice 5: (Use coloring of different types of calculations)

At times it can be useful to use colors to show different types of calculations like:

- Formulas where you use parameters as input has one color
- Where you have fixed reference in the formula (fixed col; fixed row; or fixed cell)
- Where you have formulas using input variables.
- Intermediate variables (when splitting complex formulas into sub calculations – see FGP3)
- Final output variables that have to be used in the presentation area

However, make sure that your model area does not turn into one huge color scheme so clotted with colors that you end up having less overview than intended.

Colors shall be used lightly only enhancing some few specifics giving the reader a better understanding of the transformations. Using 10 different colors and each row or formula has different colors will have the opposite effect.

Formula Good Practice 6: (Use lists as model choices)

In databases you often use some child tables to hold the different choices you can use in a specific field and you can only choose from the defined rows in these tables.

Likewise, you can build lists of choices you can choose from. Let us say that you will give the user a choice to make a budget for 2015, 2016 or 2017. You can build a model where the user shall type in the year but then the user (normally) can type in any value or you can make this as a drop-down choice where you have made a list with the 3 years in the parameter sheet with the possible domain of values.

Formula Good Practice 7: (Rounding errors and summarizations)

In many models you have to think of how you make calculations especially when you have many numbers and a sum of these numbers.

If you in the presentation area show the numbers formatted with less precision than the calculations are done you can end up having the sum of the shown detailed numbers giving another result than the summarization because of the underlying numbers have a much higher precision. At times this might be OK as you expect that the reader understands that each number is formatted with less precision than the actual underlying number but in other cases you should round each number so it corresponds with the presentation and therefore also adds up at the summary line/column.

Likewise, when you actually make some rounding then you can end up having specifications showing some details of an overview and then this is not in sync.
You can end up having a 1 or -1 difference of the underlying specification data of the output from the detailed level to the high summarization level. This should always be avoided, so try to make sure that what is expected from the detail level is also what shows up at a higher summarization level.

Formula Good Practice 8: (Avoid subtotals and totals within/middle of variable formulas)

If you calculate some variables transformations and need some subtotals or totals then do not have these in the middle of other variables. This even if this is against the idea with formulas in 'natural order' from FGP2. Only in rare cases it would be better to have summarizations 'in the middle' because of the natural order compared to have these formulas in a special area.

This is because it is better to have formulas of 'the same type' in the same area. It gives better overview and avoids errors.

Make a special area for subtotals and totals of variables.

Therefore, by taking both FGP1 and this one (FGP8) into consideration it is better to group different types of formulas than having everything in natural order. So, first group formulas by type and then in each group follow the practice of natural order.

Formula Good Practice 9: (Avoid hidden rows and columns)

Hidden rows and columns should not be used in the model sheet.

As described earlier this is often only done because you have placed the model calculations inside the presentation area.

In the table below the model area is shown from the original example in the beginning of this note:

MODEL	2010	2011	2012	comment
Calculations variables				
Revenue		11703	12218	Take Input revenue prev. Year and multiply with 1+yearly revenue increase percentage
Variable expenses		3550	3669	Take var. Exp. Prev. Year and multiply with 1+yearly exp. Increase percentage
Contribution margin	7775	8153	8549	Revenue of the year minus variable expenses of the year
Percentage variables				
Contribution margin %	69,4	69,7	70	Contribution margin of the year * 100 / revenue of the year
Index's				
Revenue index	100,0	104,4	109,0	Revenue of the year * 100 / revenue of the first year
Variable expenses index	100,0	103,3	106,8	Variable expenses of the year * 100 / variable expenses of the first year
Contribution margin index	100,0	104,9	110,0	Contribution margin of the year * 100 / contribution margin of the first year

Table 6. (Example of grouped model formulas in the Model area)

In this example we have 3 groups of calculations:

- Variables that change over time depending on input data / parameters
- Percentage calculation
- Index calculations

In this case we make sure that each area uses the same formula across columns. Each variable has a documentation comment telling how each variable has been calculated. We group first the calculations per type and then under each type use natural order from the presentation area.

Formula Good Practice 10: (Use fixed reference when needed)

When you need to copy a formula vertically (across rows) or horizontally (across columns) but you need to keep the formula fixed on a certain row or column or in some cases fixed on a certain cell then always use fixed cell references like in Excel where you use the $ sign in front of either row reference, column reference or both if you need to lock the formula to a specific cell (e.g $F12 – Col F fixed; F$12 – Row 12 fixed or F12 – Both Col F and Row 12 fixed meaning that the formula will be fixed on cell F12 when copied to other cells)

It is much better to copy formulas this way than manually has to adjust the references afterwards avoiding model errors and of course this way is much faster than manually adjusting copied formulas cell for cell.

In Excel you can set and toggle the $-sign when you point to a specific cell building the formula by pressing F4 one or more times.

Programs & Macros - Automation

Many spreadsheets (like Microsoft Excel) can handle both macros and programs.
In the first spreadsheets that was developed in the 1980'es like Lotus 123 you could record a macro where you could repeat some actions you did in the sheet. This was a special macro / spreadsheet programming language specific to the supplier/spreadsheet product.

You can still record macro's but in Excel the 'old' macro language has been replaced by the Visual Basic for Applications.

Programming Good Practice 1: (Program data import and export)

As described earlier then you should automate by either record or program (or a combination) any data import when you have to repeat data import entry on a periodic schedule. Likewise, of course you should do the same if you export data from your spreadsheet.

Programming Good Practice 2: (Make commented code)

As with all programming you should make commented code writing description of each step of the program. So, the same recommendation as described earlier with documentation of formulas.

Programming Good Practice 3: (Programming dynamics – doc. of old values)

In some case's you want your model to be dynamic like a 18 or 24 month rolling budget or other dynamics where you move everything like 1 week, 1 month, 1 year e.g. for every period depending on the scope.

In some case's you just make a copy of the old model for each period and then using a parameter you move to a new period BUT the draw-back of this method is that you do not have the historic values of your previous models in the model you work with.
You can overcome this with several methods. One of course is to build the model having all the old history in the same spreadsheet just adding new periods at the end and 'moving' all formulas accordingly.

Another approach would be to copy data (this could be only presentation data, but also in some cases model calculations and input data and variables) as values only to a special history area/history sheet. This could also be done programmatically.

General notes

General Good Practice 1: (Protecting of sheets & cells)

Many spreadsheets give the possibility to protect both sheets and cells. If you build a model where it is not allowed to change something (e.g. the model sheet) it makes sense to protect this. It is often the formulas that is needed to be protected as the validity of the model can be in danger if the users can make changes as they want.
It all depends on the usage of the spreadsheet model. If it is a one user model it is of course not as critical as if it is a multiple user model.

If you protect certain areas of a sheet where the user also shall make data entry it is suggested that you either color the input areas or in other ways make it clear for the user where data is expected to be updated.

General Good Practice 2: (Use of named ranges)

Instead of using cell references it is often possible to name certain ranges in a spreadsheet and then refer to this range-by-range name and not by sheet & cell reference.

It can be a calculation of contribution margin where the formula is written like: =Revenue-Variable Expenses.

It makes it a lot easier to understand a reference to a named area if the name is well chosen. However, only use a named range when it is obvious to identify. Use a name where that is in sync with the variable name and can be identified easily in the workbook and normally only where formulas are within the same sheet.

It is not easy to understand the model if you have calculations in the model sheet referring to named ranges in other sheets unless it is obvious that this can be found easily in the parameter sheet and/or input variable sheet.

Even though it is mentioned as a good practice to use named ranges it also has some draw backs as the viewer not directly can see if the range spans over many columns or rows which cell is actually used.

Use this General Good Practice with this in mind.

General Good Practice 3: (Use multiple spreadsheets for complex models)

If the model is very complex and/huge then it can make sense to have multiple spreadsheets to cover the different areas. It can be multiple input variable sheets (e.g. one sheet for every data entry source); multiple parameter sheets or multiple model sheets.

Handling simulations and scenarios

Simulation models and scenarios

In many cases models are used for simulations in some kind of what-if analysis setups.

This can be either because of different assumptions of the parameters of the model (different inflation rates, different currency rates e.g.) or it can be different assumptions of input variables like number of sold items or other input.

Depending on the different values of parameters and/or input variables you will have the model to calculate the different output variables in the presentation area.

Different scenarios can be handled in spreadsheets with several solutions:

1. Use different parameter values/Input variable values and save a spreadsheet per scenario
2. Use Scenario manager of What-if analysis if the Spreadsheet support this
3. Program parameter/input changes and copy output as fixed values to specific scenario area

You might have different scenarios like “Worst case”, “Realistic guess”, “Best case”. In these situation’s this can either be saved as 3 scenarios of the same model with different values or if the model should always be used in this way you might build the spreadsheet being able to handle all 3 output’s in the same spreadsheet being able to provide the values for the different scenarios as input variables and parameters and having 3 output areas either as individual presentation sheets on 3 areas on one presentation sheet.

Consider the purpose of the model and find out what is the best method to use for this kind of models.

Stochastic simulation / Monte Carlo simulation

In contrast to mathematical methods of predefined optimization formulas with deterministic algorithms like regression analysis; Wilson’s formula e.g. where you have a defined formula given specific input variables to give an optimal output, the real world more often has much more complex situations with multiple variables and parameters both with given uncertainty or different probabilities. In many of these cases you cannot setup one or a few optimal formulas to optimize the output.

When you operate with multiple different variables with different probabilities for each variable the possible different outputs can be enormous. If you have hundreds of input variables maybe with different probability distributions connected to each variable you would end up with maybe millions of possible solutions or outcome of the model.

One method is to just pick some of the most ‘possible’ input variations for different fixed scenarios as described earlier. You have to describe each scenario manually and “run” a calculation per ‘scenario’.
Another is to try to describe the probabilities for each variable and then by using random number generation to simulate different outcome for each variable/probability and you make these simulations hundreds or even thousands of times to make many different possible outputs.

The overall problem is to ‘capture’ and present the different output of the model when the model has calculated maybe thousands of different input combinations.

It is not the scope of this note to describe how to implement Monte Carlo simulation in spreadsheets but in many situations there are free or commercially available solutions with add-ins that could be much easier than building a whole model from scratch including e.g. programming of the distribution of the output variable’s.

Graphs, Charts and KPI’s in the presentation area

At times you will include some visual presentation in addition to numbers and text for the presentation area. This can be dynamic charts/graphs or some fixed graphics.

Chart Good Practice: (Graph data are grouped and placed in the model area)

With dynamic graphs that change when underlying data change you also must make sure that you do not include the underlying data in the presentation area but make sure that you make a special group/area in the model sheet (when multiple sheet layout).

Make sure that you describe where the data has come from (calculated other places in the model area, or just specific input data e.g.) and with reference to where it is used in the Presentation sheet.

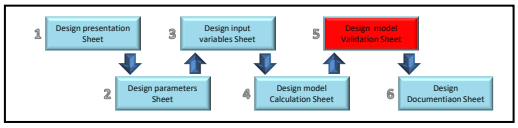
Update Presentation area with ‘+’-formulas

Finally, you should be ready to fill out the designed presentation area with data (and dynamic charts) by getting the data from the model area with the needed output variables. You do this by simply making some + formulas to transfer the output variables from the model area to the presentation area.

At times you also want to show some of the parameter choices in the presentation area and then this info is transferred in the same way. And if you also need to show some input variables these has to be transferred too.

If you use language translation of the presentation area with different language’s that can be chosen from the parameter sheet then you can either make a look-up in the parameter (or specific language sheet) sheet it-self and then transfer this with a + formula as well or you could ‘cheat’ a little and forget this purist approach with only using + formulas in this case and make a lookup formula of the language directly in the presentation area.

Step 5 - Model & Data validation



Test validation of the model

Before a model is released for production it has to be tested that the end result is as expected.

Create “script” and test data input and parameter input

If you make some test data where you manually have calculated the expected output then this can be reused when changes are made to the model.

In many cases you need to make several types of test data making sure that you cover different variants of input data and parameters.

Update the documentation sheet with version number of the model and a change log for each change and with description of test results.
For huge models and models that have to be used over and over again this step is essential. You need to prove that the model gives the results that are expected.

Fixed validation of the model

Regardless of the test and validation of the model itself there are some validation’s that should be implemented as fixed validations that could be checked every time you use the spreadsheet.

In some cases this validation is not only a question about the validation of how the model calculates but is also a validity check of the input data (see examples of this below).

Validation Good Practice 1: (Check that calculated totals equals underlying details)

This sounds trivial but this can make sense in 3 perspectives:

- Formula calculating total is wrong (taking too many or too few of detail data)
- You have some rounding issues in a total formula so total sum is different from details
- Total comes as calculated input and detail as input too, but does not give the same.

By transferring the total calculation from e.g. the model area and maybe the details from the input area and subtracting one line with the other the result should be zero as a validation check.

Try to identify all subtotals and totals used in the spreadsheet and try to make a validation that you do not have a formula problem, a rounding problem or a difference problem in the input data.

It is amazing how often e.g. rounding issues give problems so that presented detail data (maybe formatted showing less significant ciphers than actually used in the cell and therefore do not add up to the presented output calculation.

Validation Good Practice 2: (Sum horizontally and vertically gives the same)

As with checking that sum calculations is actually the total of the underlying detail data you must make sure that detail data that has a sum horizontally also gives the same sum when summarized vertically (grand total of both calculations in the intersection of the 2 sums should give the same.

Validation Good Practice 3: (Different details that should give the same must be the same)

At times you have some information that are presented in different ways but the total should be the same - Like showing revenue in a specific month per customer and revenue in the same month per product.

You must make a validation that these kind of different input of data is given the same result as expected. You can have this to be the truth when you build the model and for month after month and suddenly you have a situation because that the source input data is not valid anymore.

Validation Good Practice 4: (Check transformation - input->transformation->expected output)

If possible and you have the possibility to make some simple check of transformations where you can make an alternate check calculation to the performed transformation in the model area this should be implemented.

Some of these checks might be deterministic so that the check that you implement has to be exactly as the output model calculates like the previous checks.

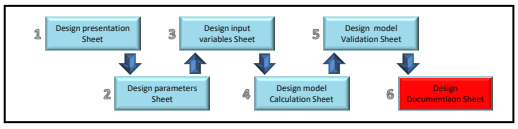
However, you could also implement some maybe not 100% certain but rules of thumbs like that the company have a policy that no sales order line can have a discount more than 25% and you have sales order lines as input data then you could calculate total discounts and make sure that these are lower than 25% of revenue before discount.

A lot of such 'sanity checks', of the input data can often be implemented.

Validation Good Practice 5: Color differences as red

If you use the suggested approach by transferring one type of data in one row/column and then what it has to be compared to in another row/column and then subtract the one with the other so the calculation shall be zero then you can use e.g. conditional formatting as in Excel to color all checks that should be zero as green and all cell's that do not equal to zero as red then you can extremely quickly identify any possible problems with the model.

Step 6 - Documentation



Different types of models

You can operate with many types of model. Figure 2 in the beginning of this note showed a visual model of the system concepts. When this note has referred to the model area in a spreadsheet it has indirectly been assumed that we talk about the transformation formulas and by this we talk about a mathematical model.

As introduced in figure 4 we have the problem that the mathematical model in a spreadsheet in general is hidden for the user. You can only see the mathematical formula in one cell at a time. Likewise as shown in figure 5 you can have huge spreadsheets with data and formulas all over.

In both cases it is difficult to create the overview of the spreadsheet. By introducing specific areas or specific sheets you can improve this overview.
By introducing comments to variables and calculations and grouping of calculations and by (lightly) using area coloring you can also improve the documentation and overview.

However, in addition to the mathematical (mostly hidden) model it is recommended, that you in addition to the split of the model into multiple different sheets add 2 additional model descriptions in the documentation sheet:

- Visual model (of the different areas and mathematical model)
- Verbal model describing the most complex and critical parts of the spreadsheet

A visual model can quickly give a new reader an overview of where the different parts are placed in the model. You should of course also make a heading of each area in the sheets used, that indicates the area like “Internal Parameters (B12-K30)” or “External Parameters (B2-K10)”, “Input Variables (B2-Z30)” etc., but a visual overview as in figure 11 and 12 below can give a very quick overview.

In addition to the visual model, then a verbal model description that explains the model logic can give a new user a quick and easy introduction to the model. Very complex calculations might be explained in more detail in a verbal model as well.

Visual model - Overview

Try to document the different areas of each sheet like in the figure 11+12 below.

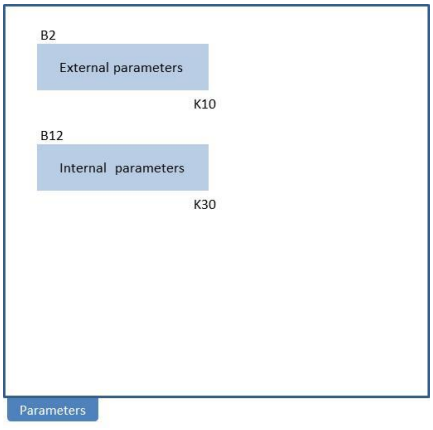


Figure 11.

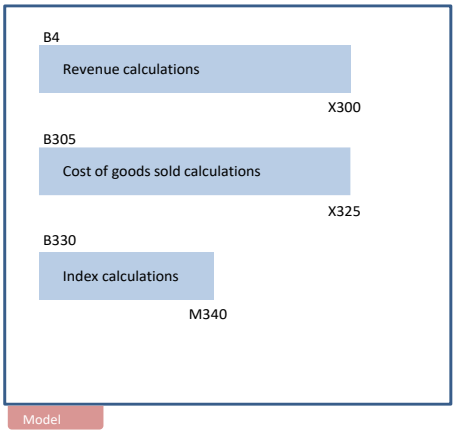


Figure 12.

In each rectangle you try to show visually write the start-cell of the area in the upper left corner and the end-cell reference in the down right corner. In this way it is very easy to understand where each part is placed within each sheet.

If you have used colors to mark specific areas in the actual sheets then it is recommended that you use the same color in the rectangles you show by the visual models.

In the examples above the overall layout of the sub-areas has been tried to be shown but in some cases you could make sub visual models describing e.g. some of the different calculation areas in some of the sub groupings also.

Verbal model - Describe formulas

In addition to the visual model showing the different areas it is recommended that you also make a verbal description of the spreadsheet.

Maybe not trying to describe every single detail of every detailed specification of e.g. formulas as it in many cases make more sense to have the description of the formula written within the model area directly with the formula itself (object principle).

However if certain overall ideas with the model; some specific important issues in regards of input variables; data import; using of parameters and so on then it makes sense to have this placed in writing directly in the spreadsheet itself as documentation (object principle again).

Try to describe the purpose of the spreadsheet. Explain the overall idea with the layout. Try to describe the model layout and grouping ideas and specific calculations or assumptions that are crucial for the spreadsheet model.

If you have programmed specific functionality like in Excel with Visual Basic for Applications you should also make a specific dedicated verbal description of what has been programmed and what it does in general (extensive commented code in the actual visual basic coding).

External documentation

It is often better to use the object principle to have the documentation saved within the spreadsheet itself so it is easy and quickly to find the documentation (and to edit and update this as the spreadsheet model changes).

However in some cases a spreadsheet is not the best authoring tool and if the model is huge and complex it might be a good idea to make additional external documentation maybe describing special functionality, programming, interfaces and so on.

Log record of versions and changes

Make sure that you have a log of changes.

Make a record of who has developed the model and document every change of the model.

Log changes to the model

Make a log of every version of the changes in the spreadsheet.

For each change there should be at least 3 entries:

What has been changed (input data, parameters, model transformation e.g., validation area)
Who has implemented the change (name, contact information e.g.)
How has the changes been tested (and with what result)

Different models – different approaches

This note has taken the approach of how to build a spreadsheet that has high transparency and easy maintenance, as well as having validated output.

However it is not always that the model is big enough or shall be used more than once or that a model builder has the needed time to implement all the recommendations suggested until now in this note.

Ad-hoc models

In many cases you only need to make an ad-hoc model for a one-time simulation and this model is not expected to be used again.

In these cases you do not need to implement documentation. Likewise you can in many cases have input variables, parameters and presentation area in the same sheet.

Minimum requirements for ad-hoc models

However it is recommended that you try to follow as many of the Formula Good Practices as possible and that you at least make sure that you keep formulas as simple as possible and rather use more rows or columns with immediate sub calculations.

Try to group formulas and do not have different types of calculations mixed so you end up not having the same formula across a row or a column (different formulas crossing each other in intersections).

Likewise that you implement as many validation rules as possible to make sure that expected results are as expected and that you try to make test data to have tested input->transformation->expected output.

Small models

For very small models you can cut back of the recommendations in the case that some of the model is trivial for most people.

You can make more in one sheet and not use the whole multiple sheet concept.

Especially documentation can be limited and in some cases you could have simple and trivial calculations directly in the presentation area.

Adjust recommendations to validation and documentation needs

In general you should adjust the ‘full recommendations in this note to the needs and type of model that has to be implemented.

This note tries to describe the full implementation with stringent model building; model validation; documentation and change log implementation but this can be modified according to the needs of the model that shall be build.