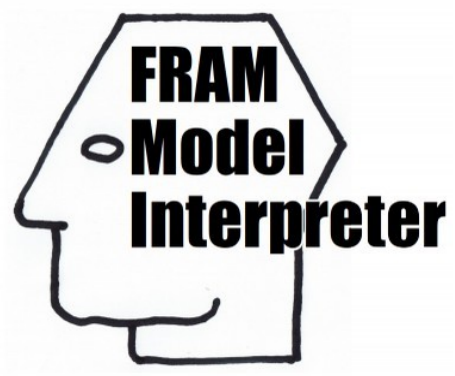


FRAM Model Interpreter (2021)

Table of Contents

Introduction

The FRAM Model Interpreter (FMI) is a software tool that can be used to interpret a FRAM model and through that determine how the described activity or task may develop. The FMI provides a realisation or interpretation of a given model in the sense that it examines the consequences of the couplings specified by the aspects of the model's functions and thereby shows how an event can develop. The FMI can therefore be used to determine how the *potential* couplings defined in the model will be realised as *actual* couplings for specified conditions – an instantiation.

Background

The purpose of the FRAM as a method is to develop a description, represented as a FRAM model, of the functions that are needed – are used, have been used, or should be used – to carry out an activity or a task taking into account how they are coupled, i.e., how they mutually depend on each other. The development of the model is most easily done by means of the FRAM Model Visualiser[1] (FMV), which is a highly effective software tool or model editor. The output from the FMV is a graphical rendering of the functions and their couplings as well as an XML file with the model details.

A FRAM model of a simple activity – making cup noodles – is shown in Figure 1. This model has a small number of functions with a limited number of couplings. Since the model is simple enough to be comprehensible from the graphical representation, it is possible to "reason" or work through it in a step-by-step dashion by following the relations or connections among functions as they are shown. Yet even this simple example makes clear that a FRAM model is different from a traditional flow model or task analysis description because there are multiple ways in which functions can be coupled, each of which has a well defined semantic identified by six different aspects (Input, Output, Precondition, Resource, Time & Control). The couplings in a model are by definition potential rather than actual and the lines that connect the functions represent possible dependencies among functions rather than a pre-defined path or flow of, e.g., control or transfer of information. Any function in a model can be described in further detail or replaced by several other functions, just as the boundary of the model can be extended by turning background functions into foreground functions. While this altogether contributes to making the model more realistic it also makes it harder and harder to "reason" through it.

1 The FMV is available through this URL: https://functionalresonance.com/the%20fram%20model%20visualiser/index.html as well as through this http://zerprize.co.nz/FRAM/index.html

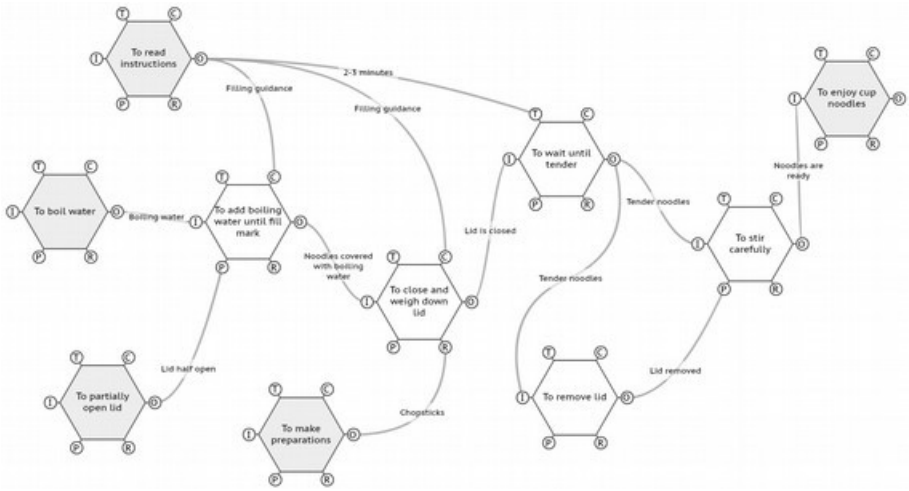


Figure 1: A FRAM model of a simple activity

Larger FRAM models, such as the one shown in Figure 2, are useful to illustrate the intricacies of an activity but it is not longer practically possible to go through them by hand, so to speak. The number of functions and in particular the number of couplings among functions prevents the kind of walk-through, talk-through analysis that is feasible for simpler models. The larger models can still be very valuable for communication, as sophisticated mind maps, and also make it possible visually to recognise possible bottlenecks or critical junctions of an activity, but they are nearly impossible to analyse in detail. It is to help overcome this problem that the FRAM Model Interpreter has been developed.

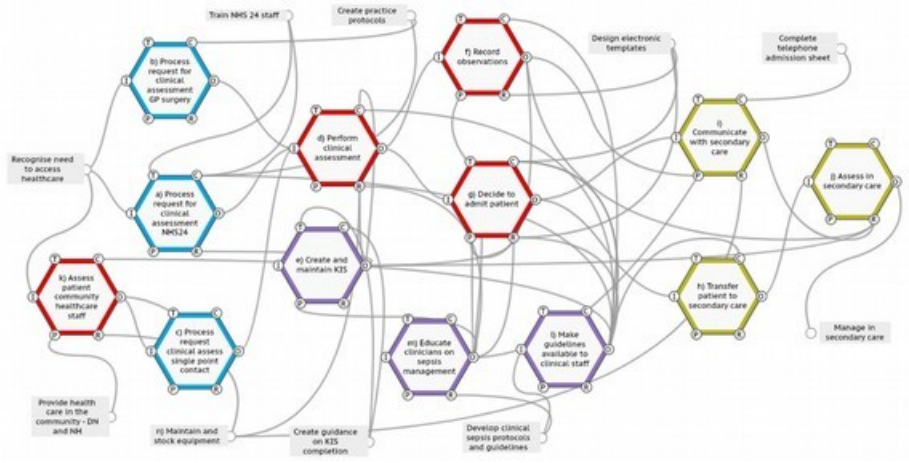


Figure 2: A FRAM model for sepsis management (McNab et al., 2018)

A Model is a Model is a Model … or is it?[2]

Models are ubiquitous in the scientific literature as well as in many other places, for instance to support a political or financial decision. Most also researchers find it irresistible to propose models as part of their work. But what is a model actually?

The two examples shown in Figure 3 both represent the main features of what we tend to call models – a set of components or entities that are connected by lines. The model components can be enclosed in boxes or other geometrical shapes or they can simply be names. The components are connected by lines that usually have a direction indicated by an arrow, sometimes even going in both directions. The components can represent different categories – for instance "effect" and "planning" in Figure 3 – just as the connecting lines can be introduced without any clear rules and represent nearly anything. The connecting lines indeed rarely have a clearly defined meaning (semantics), and the relation they represent is therefore usually ill-defined. Instead it is tacitly assumed that the reader or user can infer correctly what the connections are supposed to mean.

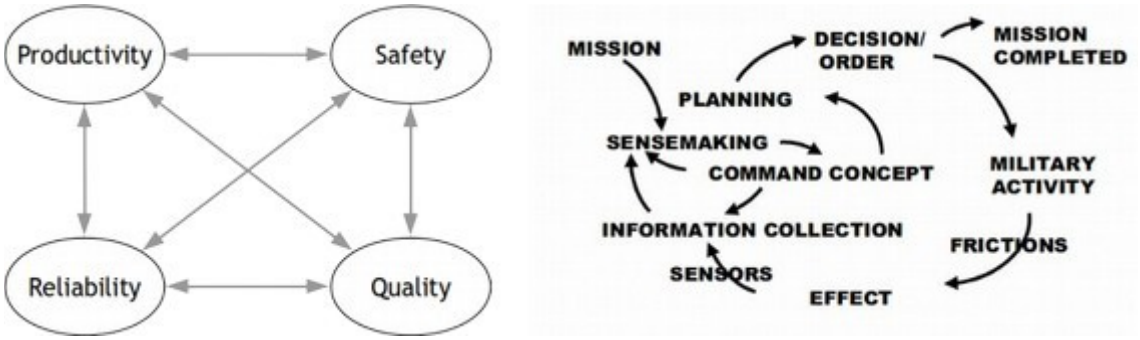


Figure 3: Two models

A model should, however, be more than a diagram of components with lines or arrows between them. The two examples shown in Figure 3 are therefore not really models, not even if the components are given more sophisticated shapes and colours. The real purpose of a model is to represent a selected set of characteristics, relations or interdependencies of something, which we can call the object system, in such a way that critical properties of the object system can be analysed in and by the model.

> "The basic defining characteristics of all models is the representation of some aspects of the world by a more abstract system. In applying a model, the investigator identifies objects and relations in the world with some elements and relations in the formal system." (Coombs, Dawes, & Tversky, 1970, p. 2)

2 With many apologies to Gertrude Stein.

A model is thus a deliberate simplification of the features or characteristics of the object system that are of interest, and the purpose of the simplification – or abstraction – is to make it easier to investigate issues of interest. The simplification must be systematic and the model must therefore be based on a well-defined set of principles that makes it possible to express or represent the "objects and relations in the world" that are being investigated. It is not enough to know **what** you want to study or represent. It is also necessary to know **how** to do it and to have a concise way of doing it. The principles that are used to build a model therefore represents the assumptions about "the world", about what lies behind the phenomena being studied or analysed. Without such principles, a model in practice becomes meaningless.

A FRAM model is a program

The FRAM is based on a set of principles – equivalence, adjustments, emergence, and resonance – that are used to describe how events can develop (equivalent to Work-as-Done) and how this may lead to intended as well as unintended outcomes. More specifically the FRAM – *qua* method – clearly specifies the basic model components and the possible relations. The components must be functions – something that is done, has been done, or can be done – either as background functions or foreground functions. And the dependencies must be in terms of one of the six aspects that are defined for the functions. The relations between functions are represented by the upstream-downstream couplings. These couplings describe how an Output from one (upstream) function may serve as an Input|Precondition|Resource|Time|Control of one or more (downstream) functions. The terms upstream and downstream refer to the relative order of functions in an instantiation, rather than to any structural features of the model.

Because of the way that the relations between functions are described, a FRAM model can actually be seen as a kind of program – as a series of instructions that control the way in which a model operates or "performs". Consider, for instance the role of the Input [Tender noodles] to the function <To remove lid> in Figure 1. This relation could also be expressed as an instruction or a piece of code in the function <To remove lid> which might look like this:

```
IF (noodles are tender) THEN remove lid
    OTHERWISE wait (i.e., do nothing)
```

Another way of stating this is that the function <To remove lid> will be activated when the state [Tender noodles] is present. In the model, [Tender noodles] is both the Output from the function <To wait until tender> and the Input to the function <To remove lid>. If the state [Tender noodles] has not yet been achieved, then the function <To remove lid> will remain in a waiting state.

As Figure 1 shows, the Input [Tender noodles] is also used as a Precondition by the function <To stir carefully>. This function therefore has a Precondition as well as an Input which both must be present before the function can be activated. The code could in this case be:

```
IF (noodles are tender) AND (lid has been removed) THEN stir carefully
    OTHERWISE wait
```

If the Input is present but the Precondition is not then the function will remain waiting. In this case the function, or rather the FMI, will "remember" that the Input has been present and wait for the Precondition to become present. Similarly, the function will also remain waiting if the Precondition is present but the Input is not. In this case the function, or rather the FMI, will "remember" that the Precondition has been present and wait for the Input to become present.

The same reasoning can be applied to every foreground function in a model. For the model in Figure 1, no function needs the presence of more than two aspects to become activated. For the model in Figure 2, the function <Decide to admit patient> needs five aspects to be present before it becomes activated, which means that the "code" for the function will be more complicated. Yet the basic principle remains: the way in which the aspects have been defined in effect constitute a small program or *method* for each function that determines when the function becomes activated. Unlike traditional programs and traditional models, there is no pre-defined sequence in which functions become activated, hence no pre-defined flow through the model. Instead each function can be seen as constantly "waiting" for the conditions that allow it to become activated, similar to the demons in the Pandemonium model (Selfridge, 1958). All that is needed is some kind of "mechanism" that can keep track of all the functions and carry out or interpret the conditions that the functions represent. The FMI is such a "mechanism" or interpretation engine.

Variability

The second of the four principles that provide the foundation for the FRAM is the principle of approximate adjustments. This principles recognised the fact that people in practically every situations will adjust what they do to match the situation – the demands, resources, opportunities, and constraints. Performance variability is inevitable, ubiquitous, and necessary. The FRAM (and Safety-II) thus changes the focus from the probability of failures – and "errors" – to the characteristics of performance variability.

In the FRAM method, performance variability is in terms of how the Output(s) from a function can vary, with variability in time and precision as the most important types. This reflects the traditional approach of safety engineering and human factors to look at the possible forms or varieties (or phenotypes) of outcomes – as illustrated by techniques such

as FMECA and HAZOP to say nothing of the many taxonomies of "human error". The variability of the output from a function can reasonably be assumed to be a result of the variability of the function or the variability of performance, but when the FRAM was developed it seemed easier to categorise types of outcomes rather than types of performance. The FMV provides the user with the option of indicating the potential variability of Outputs, but the interpretation of this is left to the user.

Variability of single aspects

One of the purposes of the FMI is to provide a way to interpret the variability of the functions in order to determine the consequences for how an activity develops, the order in which the functions become activated. Since each function can be seen as a small piece of code or program, the variability corresponds to whether these programs are executed rigorously or not. The variability is in this way resides in the function rather than in the output.[3] This is achieved by defining an **interpretation profile** for each function. The **interpretation profile** has a set of parameters that determines how the five different aspects of a function are treated.

The default assumption is that a function's method evaluates the aspects as follows:

IF ALL (defined) Inputs are present AND
IF ALL (defined) Preconditions are present AND
IF ALL (defined) Resources are present AND
IF ALL (defined) Times are present AND
IF ALL (defined) Controls are present, THEN the function is activated
OTHERWISE wait

In the FMI, the default **interpretation profile** for each function is as follows: [Input: ALL, Precondition: ALL, Resource: ALL, Time: ALL, Control: ALL]. This nominal condition can be used to show whether the activity can actually be carried out as described by the model. If that is the case, the user can then begin to change the **interpretation profile** for any foreground function. The options for each parameter are:

ALL – meaning that all aspects of the specified type must be present before the function is activated.

ANY – meaning that just one of the aspects of the specified type must be present before the function is activated.

NONE – meaning that none of the aspects of the specified type will taken into account before the function is activated.

(The options of ALL and ANY are, of course, only meaningful if there are two or more inputs to an aspect of a specified type. They are equivalent if there is only one input.

3 Relative to the FRAM it means that the propagation of variability through the upstream-downstream couplings is implicit rather than explicit.

If there are no inputs to an aspect of a specified type, the default value ALL will be equivalent to NONE.)

This can be illustrated by the "making cup noodles" model shown in Figure 1. Here the function <To stir carefully> will be activated in the nominal condition if both the Input [Tender noodles] and the Precondition [Lift removed] are present. If, however, the parameter for Precondition is changed from ALL to NONE, the function will be activated as soon as the Input is present. (The corresponding setting of the parameters for the function would be [Input: ALL, Precondition: **NONE**, Resource: ALL, Time: ALL, Control: ALL].)

The use of the **interpretation profile** can be illustrated better by considering a function with more aspects defined. An example of that is the function <To leave harbour> shown in Figure 4.

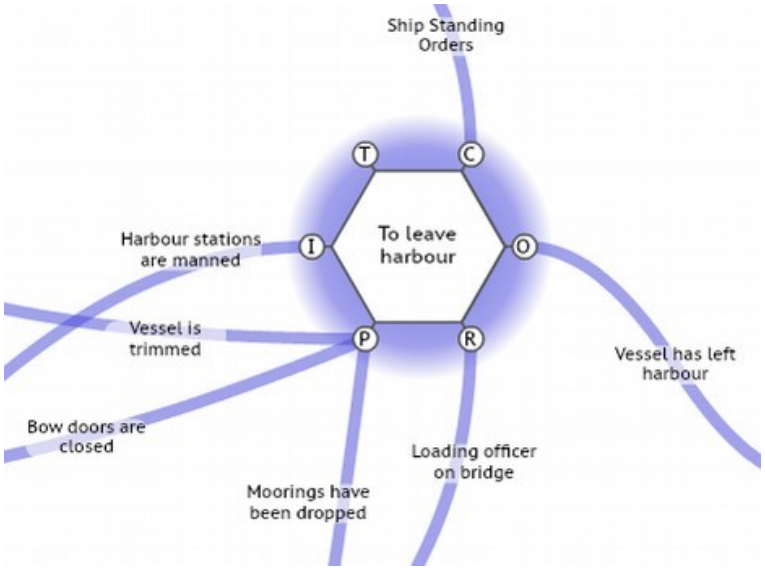


Figure 4: The function <To leave harbour>

This function has one Input, three Preconditions, one Control, and one Resource. In the nominal condition, the function will be activated if all defined aspects are present. But what would happen if some variability was defined for the function? An example could be this setting: [Input: ALL, Precondition: **ANY**, Resource: ALL, Time: ALL, Control: ALL].) In this case the function would be activated when just one of the Preconditions were present. If that, for instance, was [Moorings have been dropped] then the vessel could potentially leave the harbour without being trimmed and with the bow doors open. (Any resemblance to the *Herald of Free Enterprise* accident is intentional.)

Variability of combined aspects

In addition to specifying how a function will evaluate the aspects that have been defined for the function one by one, the user can also specify whether the evaluation will consider ALL the defined aspects or only SOME of them. This applies to the combination of four aspects Precondition|Resource|Time|Control. The combination does not include the Input because the Input must be present before the other aspects are evaluated. If that was not the case, an event could developed in a completely random manner, which would defy the purpose of making a FRAM model.

The difference between the options ALL and SOME for the combined aspects can be illustrated by the function <To leave harbour> shown in Figure 4. In this case three aspects, Precondition|Resource|Control, have been specified. Since there are three inputs to Precondition, it is possible further to specify whether the status of ANY, ALL or NONE of them is to be evaluated. Resource and Control only have one input each so ANY and ALL will be equivalent, while NONE also is a possible option.

When the three aspects are considered together it is possible to specify ALL or SOME. If ALL is specified, then all aspects will be evaluated according to the individual profile setting. If SOME is specified, then one, two or all three will be evaluated. How many is in the current version of the FMI determined by a random selection, although this option may be expanded in future versions.

Purposes of the FMI

The FMI is a multi-purpose software tool. One purpose of is to check whether the model is syntactically correct. An important part of that is the detection of orphans that the FMV has identified.[4] Other problems are potential auto-loops where the Output from a function is used directly by the function itself, or foreground functions where the specification is incomplete because there is no Input(s) or Output(s).

A second purpose of the FMI is to illustrate the difference between *potential* and *actual* couplings among functions. A FRAM model describes the *potential* couplings among functions, i.e., all the possible ways functions can be related according to how the aspects have been specified. In contrast to that, the *actual* couplings are the upstream-downstream relations that are realised when an activity is carried out, which means when the FRAM model is realised for a set of specified conditions.

A third purpose is to determine whether the activity described by the model in fact will develop as intended. In a FRAM model each foreground function defines a set of potential upstream-downstream relations through its aspects. The question is whether these relations are mutually consistent and whether they in fact will allow an event to develop as intended. It is all too easy in a complicated model to have functions that mutually depend on or block

4 In the FMV Pro, functions may be grouped and the grouping may "hide" possible orphans, i.e., they are not visible in the graphical model.

(interlock) each other, which in practice may led to conditions where functions wait forever for an aspect to become fulfilled. The FMI can identify these cases by interpreting the model step-by-step while keeping track of the status of all the aspects and activation conditions.

A further purpose is to investigate the consequences of variability of functions. In the Basic FMI this is done by means of an **interpretation profile** which specifies the conditions under which a function may become activated, rather than by considering the variability of outputs directly.

Principles of interpretation

From a programming perspective, the FMI has been developed as a production system (sometimes called a production rule system). Production systems were widely used in artificial intelligence in the 1980s and are defined as follows:

> *A production system (or production rule system) is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behaviour but it also includes the mechanism necessary to follow those rules as the system responds to states of the world.*

Leaving the pretensions of AI aside, the FMI is essentially a collection of production rules. The basic principle is that each function "looks" for the conditions that may activate or "trigger" it. These conditions include the Inputs, of course, but also the status of the aspects that have been defined for a function. If these aspects are present, the function is activated and the Output is generated. This Output will then be detected by other (downstream) functions, which then may become activated, and so on. In this way the activity is propagated through the model according to how the relations between functions have been specified, i.e., according to the potential couplings defined by the aspects.

Technically speaking, the FMI relies on asynchronous parallel execution of the functions. (It is, of course, a pseudo parallelism rather than a true parallelism.) The interpretation is asynchronous because there currently is no practical way to define a reference "clock time" – let alone real time – for a FRAM model. This means that the Time aspects only can be used to describe temporal relations between functions, such as "before", "after", or "while".

The FMI software

The FMI works as a post-processor of FRAM models and is purely text based. The FMI can read the .xfmv file that is the output from the FMV. It will parse and initialise the model to make sure that it does not include conditions that make an interpretation impossible. Once a model has been initialised an interpretation profile can be defined to control how the interpretation takes place in steps or cycles, one at a time. The stepwise

interpretation makes it possible to follow how functions become activated. The record or log of an interpretation session can be saved for later in-depth analysis.

Using the FMI software

The FMI is provided as a stand-alone software package that is available for Windows, Mac, and Linux environments. Details are provided here:

https://safetysynthesis.com/methods/fram-model-interpreter

FMI interface

When the FMI is started, the user sees the screen shown in Figure 4.

The FMI log pane on the left shows how the interpretation develops step-by-step. This information is also included in the session log. Above the pane is a field showing the name and place of the model being interpreted.

Of the two panes to the right, the upper pane shows the status of the functions in the model after each cycle of the interpretation, which the lower pane shows which Outputs are present or active after each cycle of the interpretation. This information is also included in the session log.

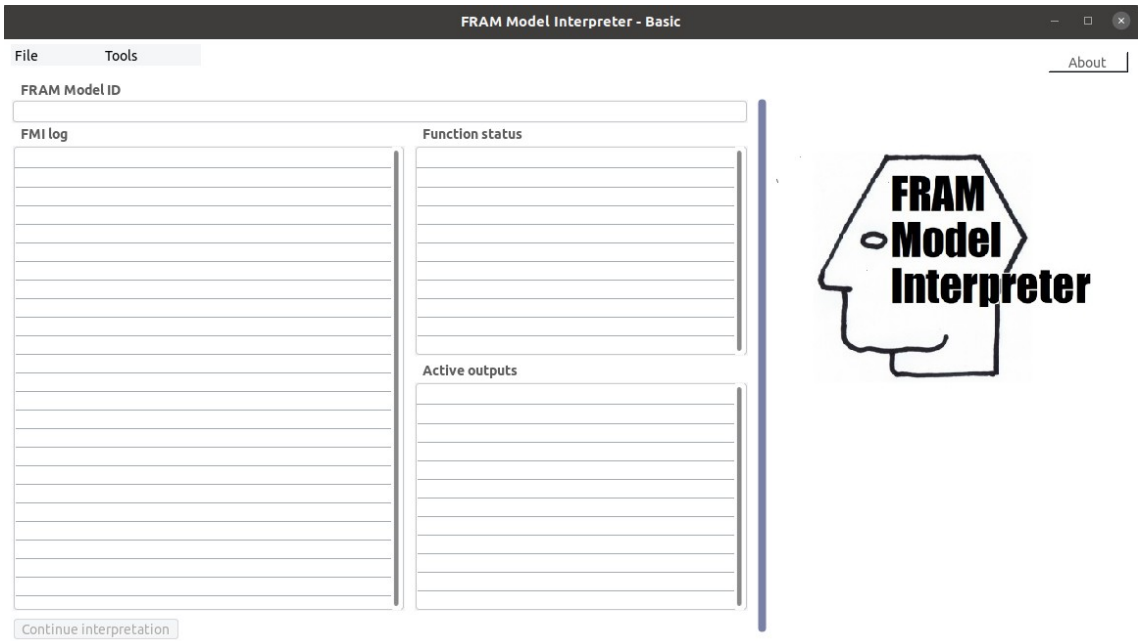


Figure 5: The FMI interface

FMI Menus

The File menu provides the following choices:

<Open model file>	The user can select which model file to open. Files must be .xfmv.

The Tools menu provides the following choices:

<Initialise>	This will parse the model and initialise it. During the initialisation the potential couplings among functions will be identified. The initialisation will also identify the Entry and Exit functions, loops, and incompletely specified functions.
	• An **Entry function** is a background function from which the Output is the Input to a foreground function. Since background functions cannot be variable, the output will be constant. It will therefore be present when the interpretation begins, which means that the downstream function can become active, provided any other specific conditions are fulfilled (i.e., precondition, resource, time or control aspects. This is, of course, only possible if there is another entry function that can provide these inputs). To prevent that the entry function continues to start the model, the Output will be cleared after the first cycle.
	• An **Exit function** is a background function where the Input comes from an upstream foreground function. When all Exit functions have been activated, the interpretation will automatically be stopped because it is not possible to take it any further.
	• An **auto-loop** exists when the Output from a function is used by the same function. Since this means that a function becomes dependent on its own Output, it can never be activated. A model that contains auto-loops cannot be interpreted.
	• A (foreground) function is **incompletely specified** if it does not produce an Output. A model with incompletely specified functions cannot be interpreted.
<Set profiles>	Selection this option will open a pane that allows the user to specify an **interpretation profile** for individual functions.
<Begin interpretation>	Selecting this option will begin the interpretation. The interpretation takes place in cycles. In each cycle all functions will be evaluated as described above and results shown in the *FMI Log*, *Function status* and *Outputs active* panes, respectively. The interpretation is continued when the ***Continue interpretation*** button is selected. The interpretation will stop when an EXIT function has been activated or if no functions were activated during the cycle.
<Reset FMI>	This is used to clear the memory after an interpretation, for instance to prepare for another model.

Interpretation Profile

An important feature of the FRAM is that functions can be variable, and that this variability may affect how the event develops – which functions become activated and in which order – and through that also the outcome.

For the current version of the FMI the variability is specified by means of an **interpretation profile** for each function. The initiation of a model will produced a default profile as described above. If the user selects the <Set profiles> option in the Tools menu, the following pane will be shown (Figure 6).

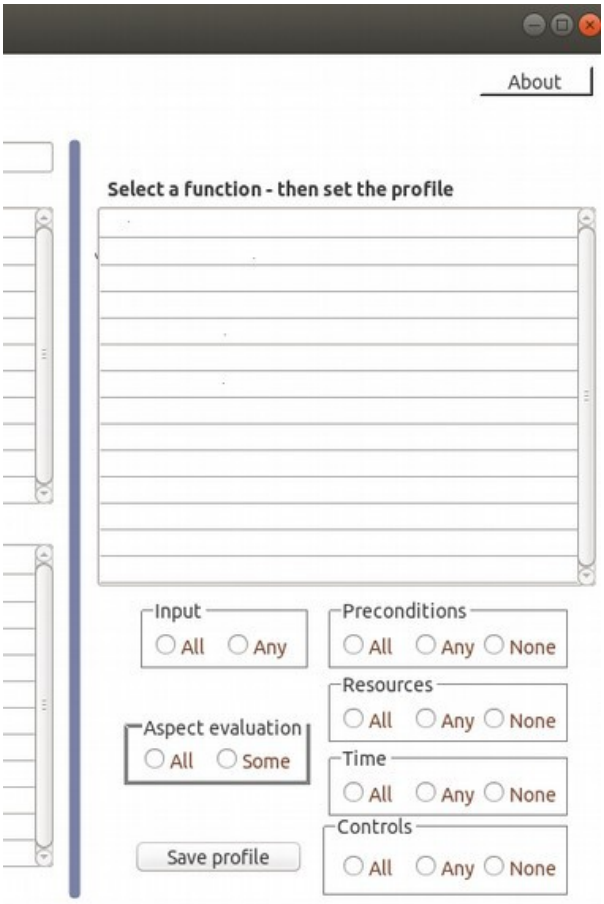


Figure 6: Interpretation Profile selection

The upper part will list all the foreground functions defined in the model. The user selects a function and then specifies the profile by means of the radio buttons.

<Input>	This applies to functions that have two or more Inputs. ANY means that the function will become READY when at least one Input is present.

	ALL means that the function will become READY only when at all Inputs are present.
<Preconditions> <Resources> <Time> <Control>	This applies to functions that have two or more aspects of the same type (P, R, T, or C). ANY means that the condition represented by the aspect will be considered fulfilled when at least one os the aspects is present. ALL means that the condition represented by the aspect will be considered fulfilled only when all of the aspects are present. NONE means that the aspect is not considered, even though it has been specified. The options for an aspect will only be shown if that aspect has been defined for the function. WARNING: It is not allowed to select NONE for all four aspects, since this would reduced the FRAM model to a simple flow model.
<Aspect evaluation>	Here the user can indicate whether the function will evaluate **all** defined aspects or only **some** of them, provided the Input(s) is/are present. Selecting SOME thus means that the function only will evaluate some but not all the aspects. Referring to the example in Figure 4, selecting ALL means that the function will evaluate the status of Preconditions, Resources, and Control while SOME only will evaluate some of them. (Time has not been defined as an aspect for this function.)
<Save profile>	Select this button to save the profile.

After the profile has been saved, the user can repeat the procedure for as many functions as needed. When this has been done, the interpretation is started by selecting the ***Begin interpretation*** item from the Tools menu.

As an example, you can use the profile to see what will happen if a function no longer checks whether resources are available (Resources set to NONE) or if only one precondition rather than all should be considered (Preconditions set to ANY). The interpretation profile is included in the session log.

A typical FMI session

In a typical FMI session, the user should proceed through the menus from left to right.

- The first step is to OPEN the model to be analysed.
- The next step is to INITIALISE the model. A model cannot be interpreted unless it has been initialised.
- An optional third step is to SET PROFILES, i.e., to specify the interpretation profiles for individual functions. All functions will initially have the default **interpretation profile**.

- The next step is to INTERPRET the model. This will be done in a step-by-step fashion. After each step the user can choose either to continue the interpretation or to stop it, based on the current state as presented in the FMI log.
- When the interpretation has come to an end, and optional step is to SAVE SESSION LOG. The contents of the log is described below.
- A final choice in the Tools menu is to RESET FMI. This can be used when the user wants to carry out another interpretation with a different **interpretation profile**, or if the user for some reason wants to start again. The reset will clear all settings for the chosen model, but not the FMV model itself. After a reset the user should repeat the selection of session log type and the initiation of the model.

FMI Session Log

The session log provides the status of all Functions and the value of all Outputs after initialisation (before cycle 0) and then after each cycle as long as the session continues. The following formats are used.

Each function is described by four items separated by tabs:

Function number	The number of the function as provided by the FMV.
Function type	The function type can be either FG (a foreground function), BG (a background function, BG Entry, or BG Exit.
Function name	The name of the function as provided by the FMV
Function status	The status can be either WAITING, READY, or ACTIVE.

Each Output is described by four items separated by tabs:

Output number	The number of the function where the Output has been defined.
Output name	The name of the Output as provided by the FMV
Output value	If the Output has not been produced during the current cycle, the value is NIL. If the Output has been produced during the current cycle, the value is the name of the Output.
Output variability	Not used in the FMI Basic version

References

Coombs, C. H., Dawes, R. M., & Tversky, A. (1970). *Mathematical psychology*. Englewood Cliffs, NJ: Prentice Hall, Inc

McNab, D., Freestone, J., Black, C., Carson-Stevens, A., & Bowie, P. (2018). Participatory design of an improvement intervention for the primary care management of possible sepsis using the Functional Resonance Analysis Method. *BMC medicine*, 16(1), 174.

Selfridge, O. G. (1958). Pandemonium: a paradigm for learning. In: *Mechanism of Thought Processes*. Proceedings of a Symposium Held at the National Physical Laboratory. (p. 513-526.)

An Example

The use of the FMI can be illustrated by the example below. This is the "Prepare cup noodles" case that often is used in FRAM courses. The FRAM model was shown in Figure 1.

After the model has been Opened and Initialised, the FMI pane looks like this:

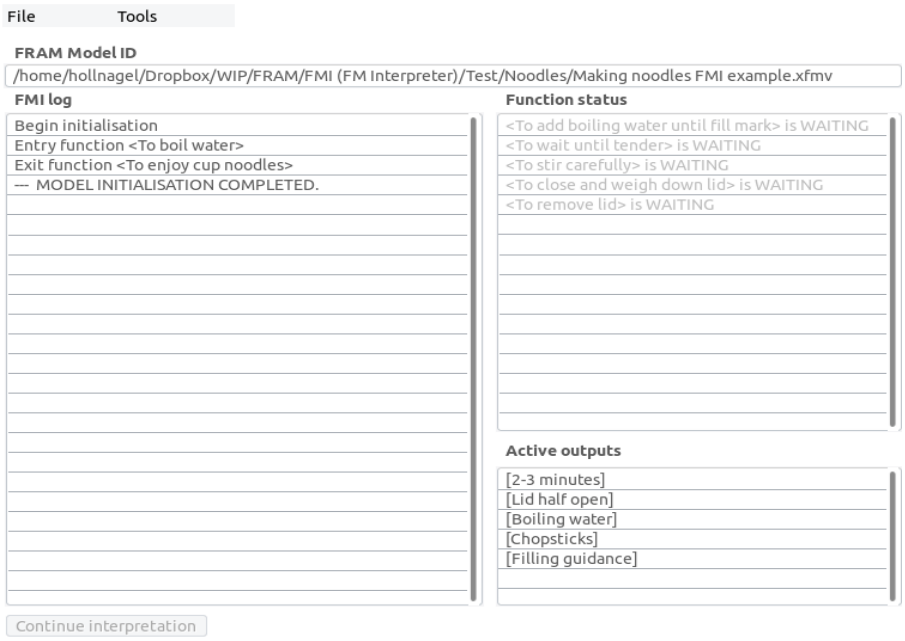


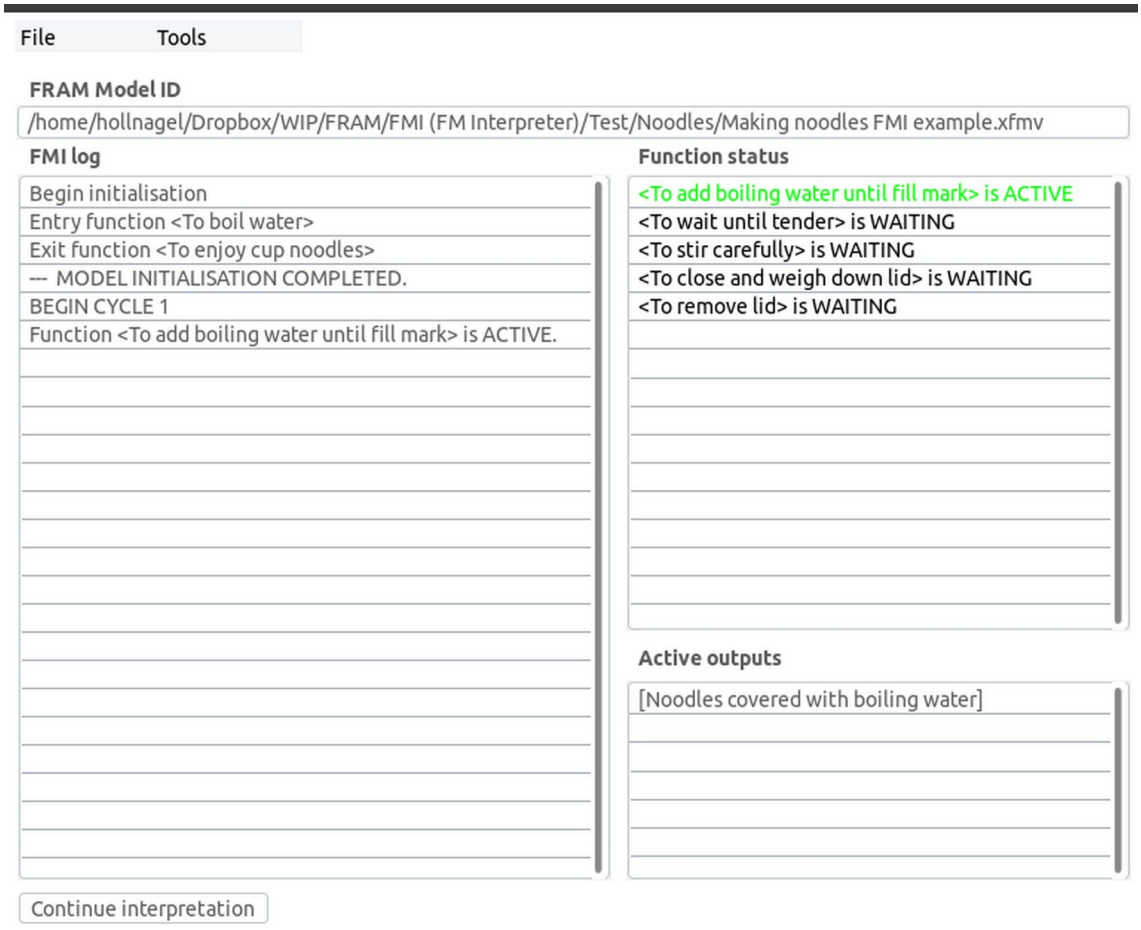
Figure 7: Noodles model after initiation in the FMI

The FMI log shows the Entry and the Exit functions that will start and end the interpretation, respectively. The model has three other background functions, but since their Outputs are not Inputs to downstream functions, they are not identified as Entry functions.

The function status pane shows the status of the foreground functions. Before the interpretation begins, all functions are WAITING.

The Active outputs pane shows the Outputs that are active at this time. Although the interpretation has not started yet, there will be Outputs that are active because they come from background functions. These Outputs will remain active throughout the interpretation, except for Entry functions where the value will be set to NIL after the first cycle has been completed.

The interpretation can now be started. After the first cycle, the FMI will look like this:



From the FMI log on the left side it can be seen that the function <To add boiling water until fill mark> has been activated because the Input has been provided by the Entry function, and because the required Precondition and Control aspects have been provided by two background functions. The corresponding Output [Noodles covered with boiling water] therefore is active. In the Function Status pane it is shown by colouring the function green. As the interpretation is continued, the FMI log will show step by step which functions have become activated and which Outputs are active. The activated functions will also be coloured green in the Function Status pane.

In cycle 2, <To close and weigh down lid> is activated because the Input has been provided by the activation of <To add boiling water until fill mark>, and because the required Resource and Control aspects have been provided by two background functions.

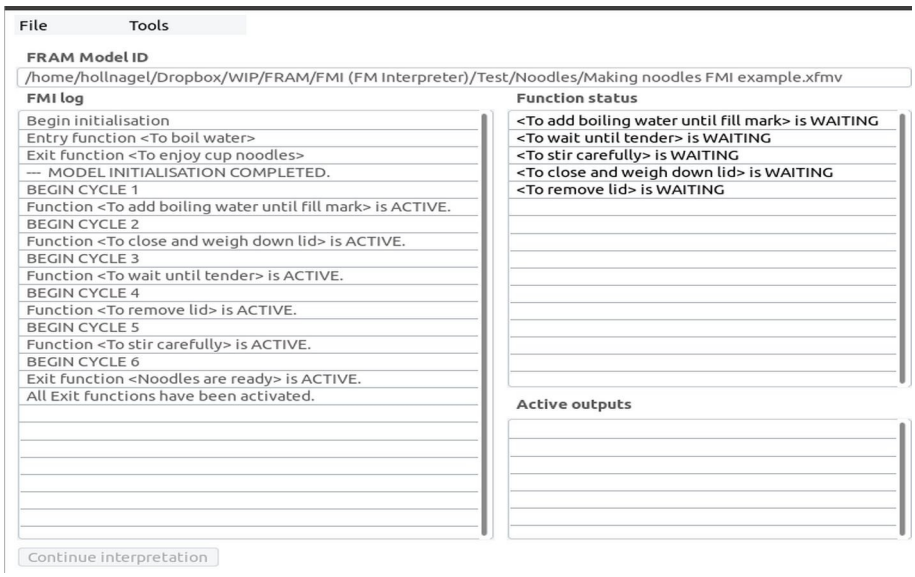
In cycle 3, <To wait until tender> is activated because the Input has been provided by the activation of <To close and weigh down lid>, and because the required Time aspect has been provided by the background function <To read instructions>.

In cycle 4, <To remove lid> is activated because the Input has been provided by the activation of <To wait until tender>. There are no other aspects defined for this function.

Notice that <To stir carefully> is not activated during this cycle even though the Input to that functions also has been provided by the activation of <To wait until tender>. There is, however, a Precondition which is the Output from <To remove lid> but the detection of this Precondition must wait for the next cycle. The function <To stir carefully> "remembers" that the Input has been provided as long as required. In the FMI all functions have a "local memory" for the value of their aspects; this "local memory" is cleared once a function is activated. The contents of the "local memory" is documented in the Detailed form of the Session Log.

In cycle 5, <To stir carefully> is activated because the Input was provided by the activation of <To wait until tender> in cycle 2 and because the required Precondition was provided by the activation of <To remove lid> in cycle 4.

In cycle 6, the Exit function is activated and the interpretation is therefore completed.



Conditions of use

The FMI software is provided "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the developer be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

The FMI software is provided free of charge and must not be sold for commercial purposes in either the original or a repackaged form.