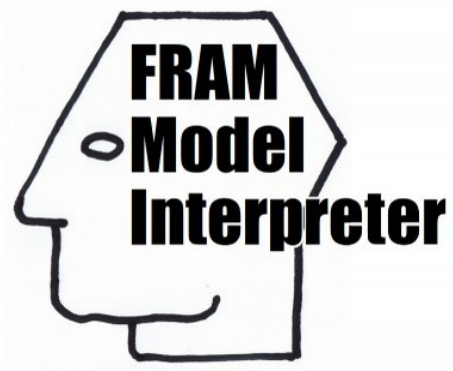

FRAM Model Interpreter (MMXXII)

Table of Contents

Introduction

The FRAM Model Interpreter (FMI) is a software tool that can interpret a FRAM model. The FMI provides a realisation or interpretation of a given model in the sense that it systematically examines the consequences of the couplings specified by the aspects of the model's functions and thereby shows how an event can develop depending on the variability of the functions. The FMI can thus 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. A FRAM model takes into account how the activities are coupled, i.e., how they mutually depend on and affect 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 which contains 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 of which some are coupled to others. 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 fashion by following the relations or connections among functions as they can be seen. Yet even this simple example clearly shows that a FRAM model is different from a traditional flow model – e.g., as the result of a task analysis – because there are multiple ways in which functions can be coupled. Each coupling has a well defined meaning referring to one of 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 energy or information. Any function in a model can, of course, be described in further detail by replacing it by several other functions. The boundary or scope of the model can also be increased by turning background functions into foreground functions or decreased by replacing foreground functions by background functions. While increasing the resolution of the model makes it more realistic it also makes it increasingly difficult to "reason" through it.

1 The FMV is available from http://zerprize.co.nz/Home/FRAM

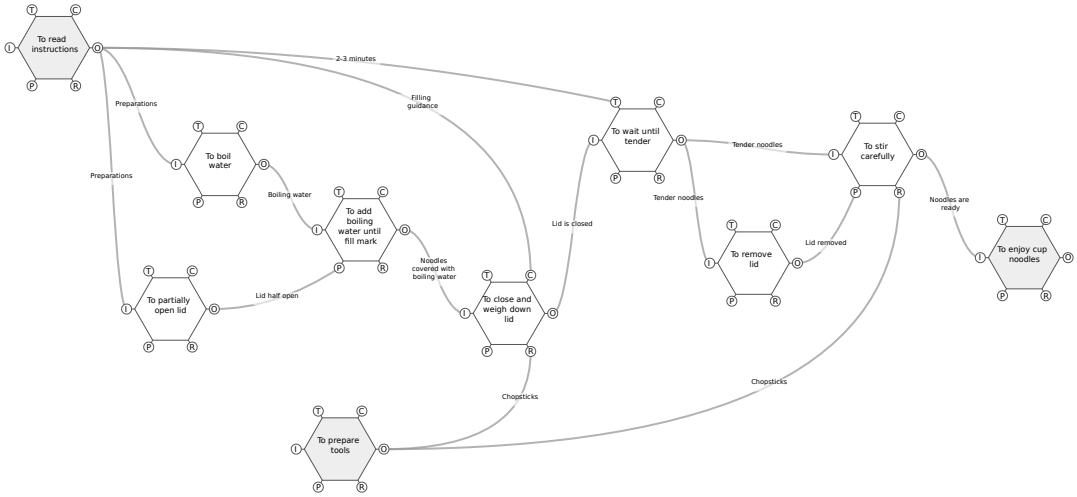

Figure 1: A FRAM model of a simple activity

Larger FRAM models of more intricate activities (Figure 2), which typically also require the collaboration and coordination of several people, are useful to illustrate the intricacies of an activity but it is not 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. They 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.

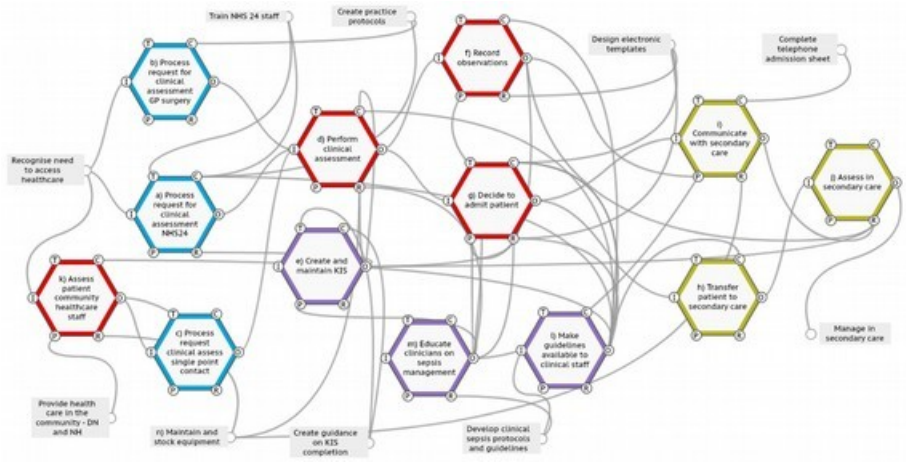

Figure 2: A FRAM model for sepsis management (McNab et al., 2018)

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 political or financial decisions (Borner, 2021). Most researchers also 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 essential features of what we commonly 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 can be 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 relations they represent are therefore usually ill-defined. In most cases it is tacitly assumed that the reader or user can infer correctly what the connections are supposed to mean.

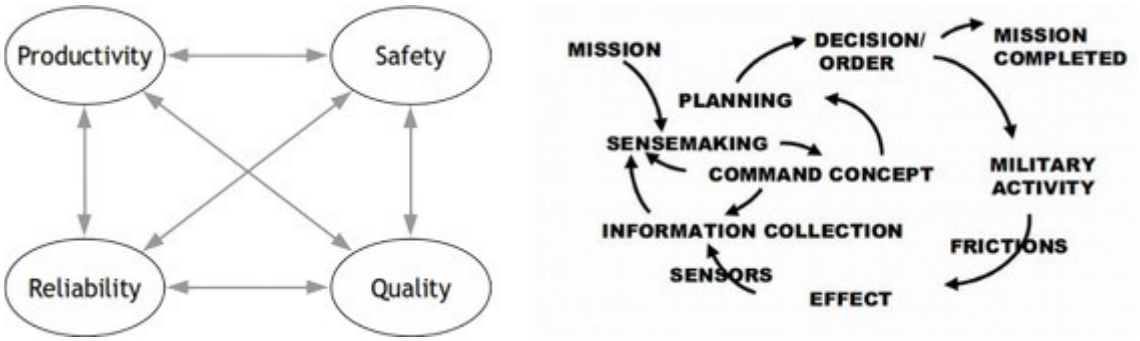

Figure 3: Two models

A model should, however, be more than a geometrical 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 (called the object system) in such a way that critical properties of the object system can be analysed by means of the model (or the model system).

> "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

2 With many apologies to Gertrude Stein.

> objects and relations in the world with some elements and relations in the formal system." (Coombs, Dawes & Tversky, 1970, p. 2)

A model is thus a deliberate simplification of 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 or analyse issues of interest. The simplification must be systematic and the **method** by which the model is produced must be based on a well-defined set of principles. These principles determine how to express or represent the "objects and relations in the world" that are being investigated and ensures that the resulting model is a product of the method rather than of the modeller's skills. 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 four principles as a basis for how to build a model of an activity.[3] The model represents functions – something that is done, has been done, or that can be done – either as foreground functions or background functions.[4] The mutual dependencies are represented by 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.

The relations between functions can be used to generate the rules that determine when a function is activated. Because of that a FRAM model can 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 (Tender noodles IS TRUE) THEN remove lid
    ELSE wait
```

3 The four principles of the FRAM are the principle of equivalence of successes and failures, the principle of approximate adjustments, the principle of emergence, and the principle of functional resonance (Hollnagel, 2012).

4 In the FRAM foreground (FG) functions can be variable whereas background (BG) functions cannot.

In this case the function <To remove lid> will be activated when the Input [Tender noodles] is true. In the model, the aspect [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 (Tender noodles IS TRUE) AND (Lid Removed IS TRUE) THEN stir carefully
    ELSE 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 "examining" 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 of functions

The second of the four FRAM principles 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 or malfunctions to the characteristics of performance variability.

The FRAM method initially represented performance variability indirectly by how the Output(s) from a function could vary, with variability in time and precision as the most important types. This reflected 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 was assumed to be a result of the variability of how the function was carried out, hence an expression of performance variability.

A better solution is to use the variability of the functions directly to determine when a function will be activated, hence how an activity develops. Since each function can be seen as a small piece of code, the variability corresponds to differences in how the five different aspects of a function are evaluated. This is achieved by means of the **control mode** for each function. (A detailed description of the underlying theory is provided by Hollnagel & Woods, 2005.)

The control modes represent the orderliness or regularity of how a function is carried out. Four different control modes are defined as follows:

- In the **strategic control mode** there is by definition no variability. The strategic control mode corresponds to a condition where there is sufficient time – and resources – thoroughly to consider every aspect that has been defined for a function. Any variability in the Output from an upstream function will, of course, remain and therefore be propagated further, but no new variability will be introduced.
- In the **tactical control mode** there can be some variability in how a function is activated. This means that only some but not all (defined) aspects will be considered for Resources, Time and Control. This may also have an effect on the variability of the Output.
- In the **opportunistic control mode** there is even more variability in how a function is activated. This happens by reducing the thoroughness by which the defined aspects are considered. This will, of course, also have an effect on the variability of the Output.
- Finally, in the **scrambled control mode** a function is activated whenever an Input is present. None of the other aspects are taken into account, and the Output will be highly variable.

The characteristics of the evaluation of the aspects in the four control modes is shown in Table 1 below. (Table 1 only defines the relation between the control mode and the evaluation of the aspects. The description of how the control mode affects the Output(s) from a function will be added later.)

Table 1: Control mode characteristics

	Evaluation of aspects defined for the function				
Control mode	Input	Preconds	Resources	Time	Control
Strategic (STRA)	ALL	ALL	ALL	ALL	ALL
Tactical (TACT)	ALL	ALL	ANY	ANY	ANY
Opportunistic (OPPO)	ANY	ANY	ANY	NONE	NONE
Scrambled (SCRA)	ANY	NONE	NONE	NONE	NONE

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

As an illustration of how the control modes affect how a function is activated, a strategic control mode means the following:

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, THEN the function is activated
ELSE wait

An opportunistic control mode will look like this (the Time and Control are not considered at all):

IF ANY (defined) Inputs are present AND
IF ANY (defined) Preconditions are present AND
IF ANY (defined) Resources are present THEN the function is activated
OTHERWISE wait

In this case the function will be activated if at least one of the Inputs and at least one of the defined Preconditions and at least one of the defined Resources are present, whereas the Time and Control aspects will not be considered at all. This corresponds to a

condition where there is less thoroughness in carrying out an activity, which is a not uncommon form of performance variability.

The use of the **control modes** can be illustrated by considering a function for which many aspects have been defined. An example of that is the function <To leave harbour> shown in Figure 4.

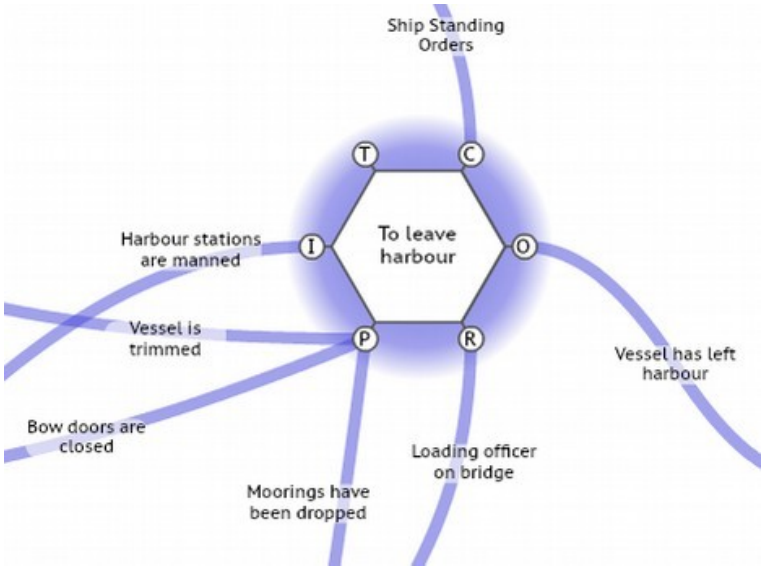

Figure 4: The function <To leave harbour>

This function has one Input, three Preconditions, one Control, and one Resource. In the strategic control mode, the function will be activated if all defined aspects are present. But in a tactical control mode, the activation conditions would be like this: [Input: **ALL**, Precondition: **ALL**, Resource: **ANY**, Time: **ANY**, Control: **ANY**].) In this case the vessel could potentially leave the harbour without the loading officer on the bridge and without observing the Ship Standing Orders. (Any resemblance to the *Herald of Free Enterprise* accident is intentional.)

Purposes of the FMI

The FMI is a multi-purpose software tool which serves the following purposes.

Syntax check

One purpose is to check whether the model is syntactically correct. There are several conditions which, if present, makes it impossible for a model to be interpreted.

- **Orphans**: An orphan is an aspect that only is defined for one function. In order to make sense, every aspect must be defined as an Output for (at least) one function and

an [I, P, R, T, or C] aspect for at least one other function. The FMV automatically identifies aspects, and the FMI simply repeats the test.[5]

- **Auto-loops**: An auto-loop is a condition where the Output from a function is used directly by the function itself.
- **Passive foreground functions**: If a foreground function does not have an Input defined, it will never be executed. If a foreground function does not have an Output defined, it does not produce or create anything. In both cases the foreground function is passive, i.e., it plays no role in the model.
- **Permanent preconditions**: A Precondition represents a variable condition that must be present before a function can be activated. If the Precondition is linked to the Output from a background function, it means that the Preconditions always is fulfilled. Since this is meaningless, it is classified as a syntactical error.

Potential and actual couplings

The FMI can be used 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.

The FMI can also be used to show 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 will ensure that an event develops as intended. It is all too easy in a complicated model to have functions that mutually depend on or block (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.

The FMI can finally be used to investigate the consequences of variability of functions by setting the **control mode** as explained above.

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:

5 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.

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 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 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 is used 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, the control mode can be specified for any foreground functions. This will determine how the interpretation takes place step by step. 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. The FMI software as well as this user guide are available for download here:

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

FMI interface

When the FMI is started, the user will see the screen shown in Figure 5.

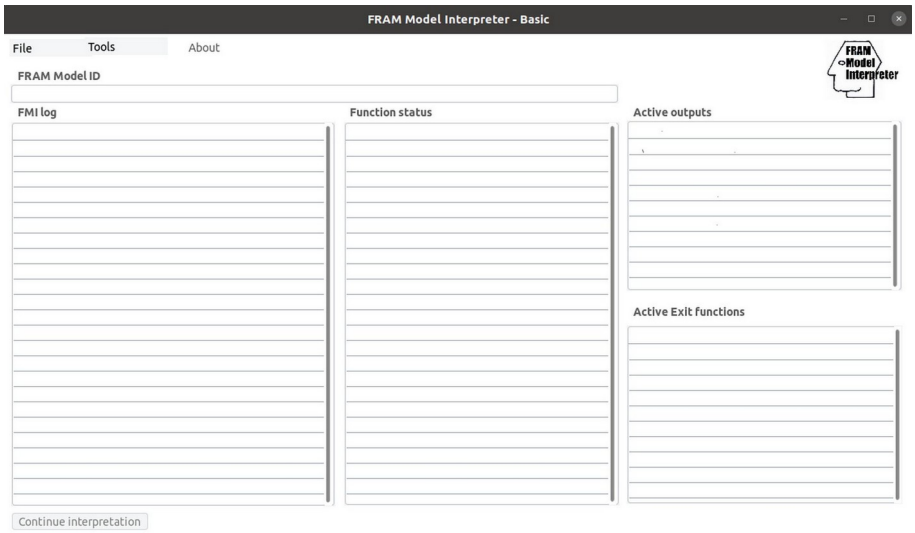

Figure 5: The FMI interface

The **FMI log** pane on the left will show the interpretation progress 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.

The **Function status** pane in the middle shows the status and control mode of the functions in the model after each cycle of the interpretation. Functions that have been activated in the current step of the interpretation will be shown in green colour. Of the two panes to the right, the upper shows the **Active outputs** after each cycle of the interpretation, while the lower shows the **Active Exit functions**. All this information is also included in the session log.

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, passive foreground functions, and permanent preconditions.
	• 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 in the second and following cycles, 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.
<Set control mode>	Selection this option will open a pane that allows the user to specify the **control mode** 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 *Active outputs* panes, respectively. The interpretation is continued when the ***Continue interpretation*** button is selected. The interpretation will stop when all EXIT functions have been activated or if no functions were activated during the cycle.
<Reset>	This is used to clear the memory after an interpretation, for instance to prepare for another model.

A typical FMI session

In a typical FMI session, the user should proceed as follows.

- The first step is to OPEN the model to be analysed using the File menu.
- The second step is to INITIALISE the model. A model cannot be interpreted unless it has been initialised first. The initialisation will identify the **Entry functions** and **Exit functions** and list these in the **FMI log**.
- An optional third step is to SET CONTROL MODE, i.e., to specify the **control mode** for individual functions. The default control mode for all functions is Strategic (STRA).
- The next step is to INTERPRET the model. This will be done in a step-by-step fashion. After the first step the user can continue the interpretation by clicking the **Continue interpretation** button.
- The interpretation will automatically end when all **Exit functions** have been activated. When this happens, the user can choose to save the session logs. The contents of the logs are described below.
- A final choice in the Tools menu is to **Reset** FMI. The reset will clear all settings for the chosen model, but not the FMV model itself. After a reset the user may repeat the initiation of the model.

Set Control Modes

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.

In the FMI the variability can be specified by setting the **control mode** for each foreground function. The default setting for all functions is the Strategic control mode. If the user selects the <Set control mode> option in the Tools menu, the Active outputs pane will be replaced by the Control mode pane as shown in Figure 6.

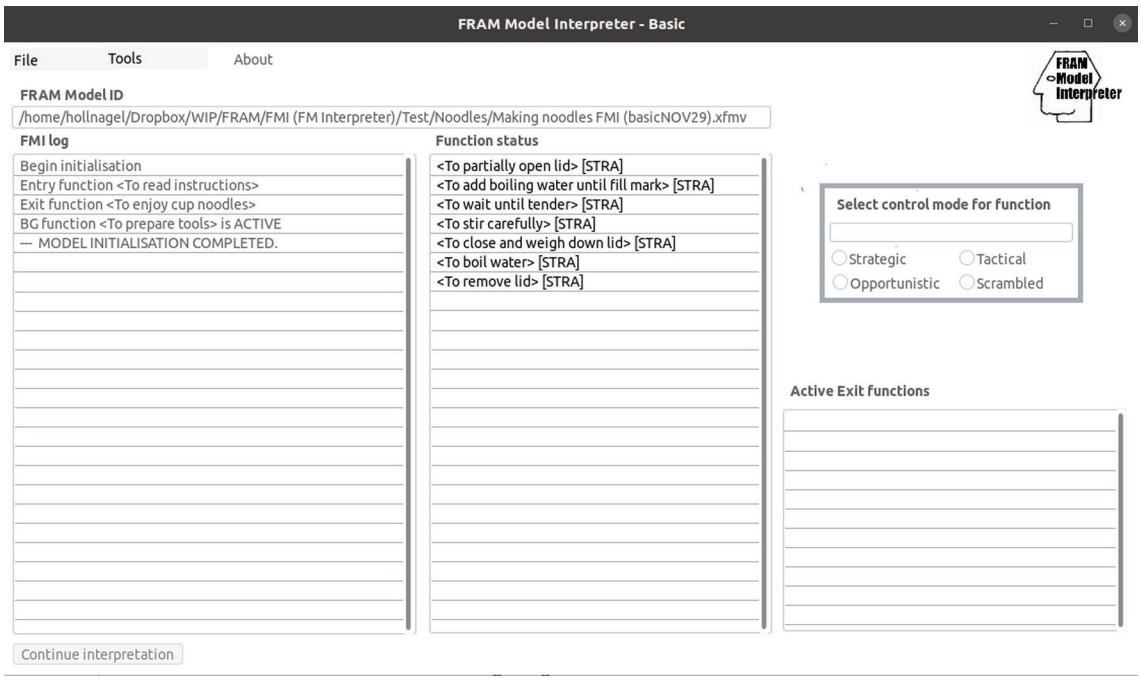

Figure 6: Control mode selection

The **Function status** pane will list all the foreground functions defined in the model and also their current control model (which by default is STRA). To default control mode can be changed as follows:

Select a function. The function will be highlighted in the **Function status** pane. The name of the function will be shown in the **Select control mode** pane and the current control mode will be shown by an active radio button (cf., Figure 7).

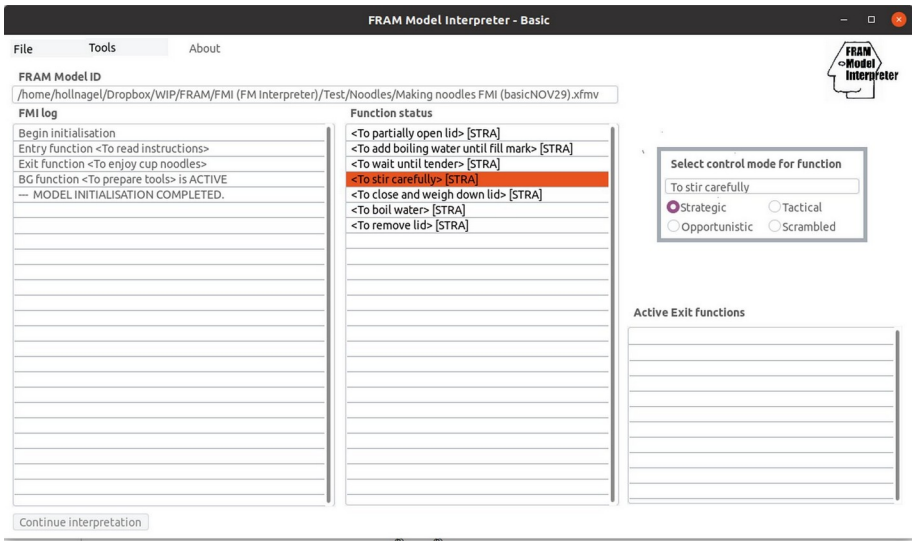

Figure 7: Selecting a function

The user can change the control mode of the function by using the radio buttons. The result of that will be shown both as an entry in the **FMI log** and in the **Function status** pane. (Figure 8 shows the result of changing the control mode for <To stir carefully> from STRA to OPPO.)

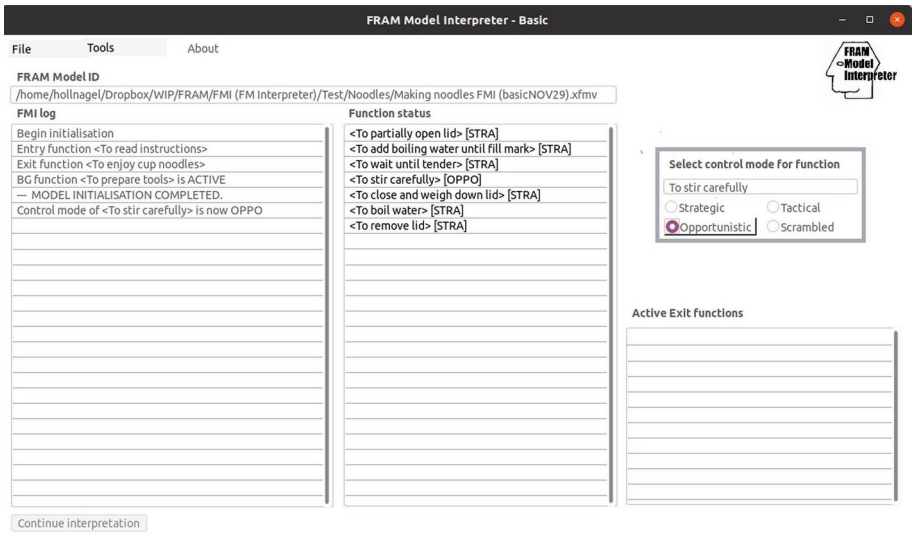

Figure 8: Changing the control mode of a function

When the user has made all the necessary changes, the interpretation is started by selecting the **Begin interpretation** option from the Tools menu. When the interpretation begins both the **Initialise option** and the **Set control mode** option will become inactive.

Begin interpretation

When the **Begin interpretation** option has been chosen, the FMI interface will revert to the general state shown in Figure 5. The first step of the interpretation will also be carried out. The second and following steps will be carried out by clicking the **Continue interpretation** button.

After each step, the FMI interface will show the results in the following way (cf., Figure 9).

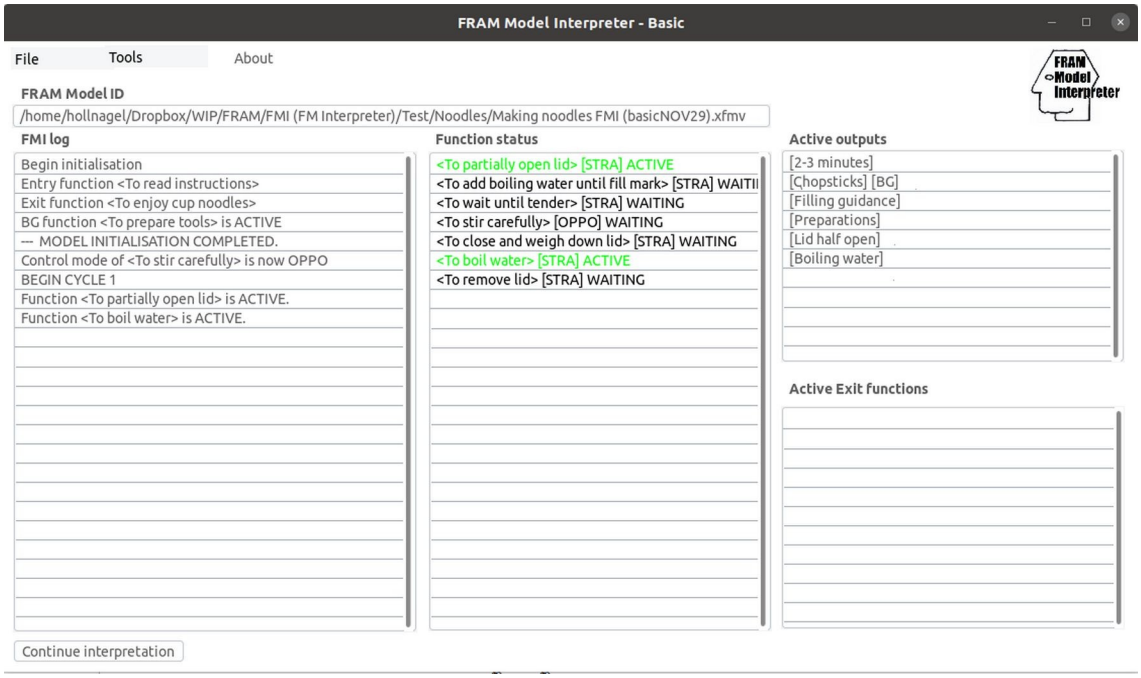

Figure 9: The FMI display after an interpretation step

The **FMI log** will show the number of the cycle (as "BEGIN CYCLE *n*") and list the functions that have become active during this cycle. The functions will also be shown in green in the **Function status** pane. The **Active outputs** pane will show the Outputs that are active after the current step. Finally, the **Active Exit functions** pane will show the Exit functions, if any, that have become active.

Figure 9 shows the status after the first interpretation step of the FRAM model of making cup noodles (Figure 1). In this case there is one **Entry function**, <To read instructions> and one **Exit function**, <To enjoy cup noodles>. The initialisation has also identified a permanent background function <To prepare tools> which is not an Entry function.

After the first cycle the functions <To partially open lid> and <To boil water> have become activated. This is shown both in the **FMI log** and by a change of colour in the **Function status** pane.

There are also six Outputs that are active after this cycle, as shown in the **Active outputs** pane. One of these [Chopsticks] comes from a permanent background function <To prepare tools>. Three of the Outputs, [2-3 minutes], [Filling guidance] and [Preparations], come from the Entry function <To read instructions>. And two Outputs come from the functions that became active during this cycle. The Output [Boiling water] comes from the function <To boil water> and the Output [Lid half open] comes from the function <To partially open lid>.

No **Exit functions** became active during this cycle.

The interpretation can now be continued by clicking the **Continue interpretation** button after each step. The interpretation will continue either until all Exist functions have been activated, or until if there has not been any change after a cycle. (This may happen if the model includes an impossible condition, for instance that a Precondition to a function is the Output of a downstream function that is not a neighbouring function.) In the present example, the final screen is shown in Figure 10.

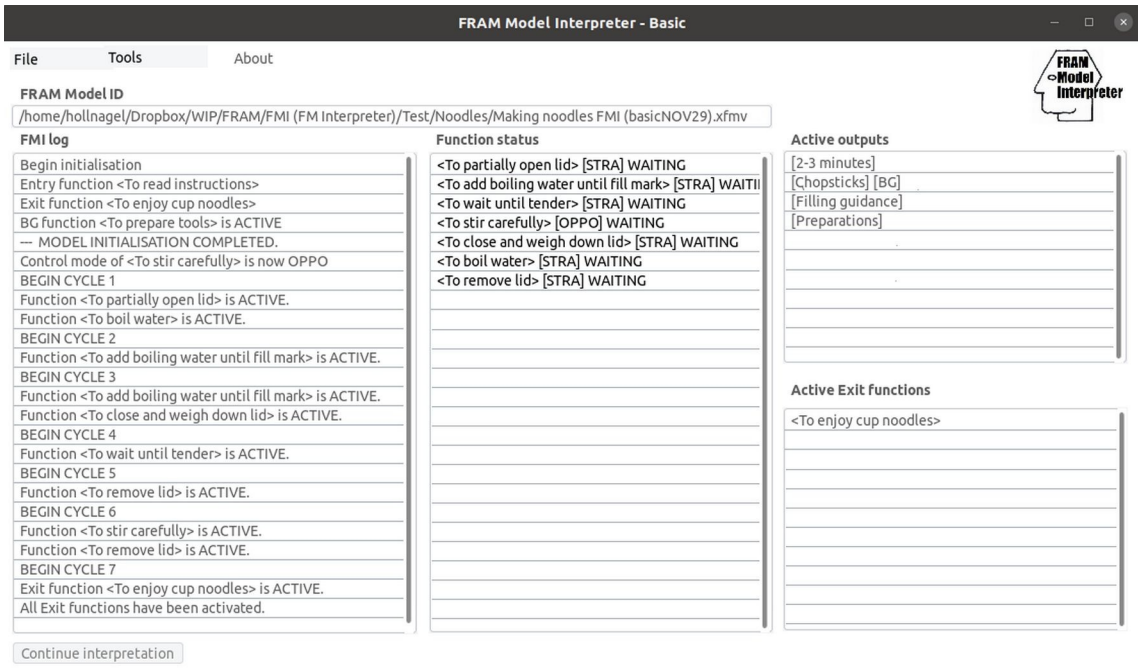

Figure 10: Status after final interpretation cycle

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

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.

References

Borner, K. (2021). *Atlas of Forecasts: Modeling and Mapping Desirable Futures*. MIT Press.

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

Hollnagel, E. (2012). *The Functional Resonance Analysis Method: Modelling complex socio-technical systems*. Farnham, UK: Ashgate.

Hollnagel, E. & Woods, D. D. (2005). *Joint cognitive systems: Foundations of cognitive systems engineering*. Boca Raton, FL: CRC Press / Taylor & Francis.

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.)