**Session One**

SLIDE 3 (Purpose and Scope)

Introduce the Session. The next Slide represents the start of the formal evaluation, and to maintain a uniform approach with each participant, the tutorial which provides introduction to the framework and the use of FRAM follows a written script. There will be time for any questions at the end of the tutorial.

As we request your feedback on the clarity of the written process instructions, no consideration of the process steps is made in this tutorial.

SLIDE 5 (Framework)

We have created a framework in support of the process by which an organisation can model, understand, and assess software safety practice. The framework asserts the disparate elements that constitute software safety practice, the relationships between the elements, and denotes the comparisons required by the process.

The aim of the framework is to enable an understanding of how software safety practice is carried out, and why it is done in the manner that it is. Specifically, we seek an understanding of how software safety practice is desired to be, how the desired practice is imparted to those required to enact it, and how software safety practice as desired is interpreted and implemented by software safety practitioners.

The framework presented here is a combination of modelling and comparison activities. There are 4 software safety modelling tasks (Steps 1 to 4), with each modelling representing a different element of software safety practice. The elements of practice modelled in this framework are:

* As-desired – how software safety practice is required to be
* As required (Open Standard) – the practice represented in functional safety standards
* As required (Closed Standard) – the software safety practice as expressed in an organisation’s safety lifecycle process documents…AND
* As observed – the practice carried out by software safety engineers, or software engineers with responsibility for functional safety.

There are also 6 comparison activities which require already-created models to be assessed and compared. These comparison activities will be discussed in the second evaluation session, however.

Each step of the process has an accompanying set of process instructions, and you will be asked to follow some of these steps as part of the overall evaluation process.

SLIDE 6 (Introduction to FRAM - Original FRAM):

To understand software safety practice fully, we need a means to discuss and  
evaluate practice in a robust manner that allows the different elements of practice to be represented equally. After undertaking research not discussed here, we have selected a tool called FRAM to represent practice. The tutorial now introduces FRAM, and how we have modified it for the purpose of this research.

FRAM stands for the Functional Resonance Analysis Method. Originally, FRAM was used to model a successful outcome and measure the resonance (or the impact) of changes. As you can see on the screen, FRAM is constituted by either activities or functions - each of which has 6 quality attributes – referred to as aspects. The aspects represent:

* Its inputs
* The time by which an activity should occur
* The means by which an activity is controlled
* Its outputs
* The resources consumed by the activity
* …and any preconditions which must be met for the activity to proceed/complete.

SLIDE 7 (Introduction to FRAM - Original FRAM Example)

The example of FRAM on this slide shows the abridged activities required to withdraw cash from an ATM. In this original FRAM notation, preconditions are modelled using sub-activities which are shaded in grey.

In this example, the customer requiring cash must first insert their card, and then enter their PIN. The entering of a PIN is a security check, and the precondition for entering the PIN is the user’s ability to recall it. After entering the correct PIN, the customer specifies the amount to withdraw before collecting it – as long as they have sufficient funds in their bank account.

SLIDE 8 (Modified FRAM – changes to Aspects)

Whilst retaining most of the original ontology and symbology of FRAM, we have made small modifications to FRAM for this research, and we model only activities – not functions. In terms of the aspects of an activity, the only change we have made is to substitute ‘Preconditions’ for ‘Method or Technique’ (the aspect denoted ‘M’) – and this is designed to model the technique or method by which an activity is performed. In our version of FRAM – FRAMSP (for safety practice), Preconditions are managed by representing the required inputs to an activity.

SLIDE 9 (Modified FRAM – Aspects of Activities)

In FRAMSP, all aspects are stated as a noun, or noun-phrase. Inputs are those used or transformed by an activity to produce an output, or items which activate an activity. Outputs describe the result of the activity, and an output from one activity will always be used as an Input, Method, Resource, or Control to another activity pertinent to software safety practice.

SLIDE 10 (Modified FRAM – Aspects of Activities Cont’d)

Resources are needed and consumed by an activity. Controls supervise or regulate an activity so that it produces the desired output (and may therefore be a quality stipulation). Time denotes the temporal relations which may affect when an activity is carried out; and it can relate to elapsed/clock time; a date; a phase; or relate to a point within a sequence of operations.

SLIDE 11 (Modified FRAM - Introducing Artefacts)

At some point in a lifecycle model, we reach the lowest level of abstraction required – where no further modelling may be necessary (from a software safety practice point of view at least). Artefacts are items consumed or produced by an activity in the safety lifecycle. Modelling 6 aspects for artefacts (as we do for activities) would be overly-onerous and superfluous, and so we only model 3 aspects for an artefact – Time, Quality Criteria, and Existence. Existence refers to whether the artefact already exists – and if not whether an activity needs to be carried out by the software safety team to produce it. The quality criteria of an artefact is wide-ranging and could be the skills and experience required of the person charged with carrying out an activity, or perhaps the format and contents required of a report.

SLIDE 12 (Modified FRAM – Aspects of Artefacts)

To recap the aspects of artefacts then. Time expresses the point by which an artefact should be created by, or supplied to an activity. Quality Criteria is used to define the quality attributes required of an artefact – such as the required skills or experience of a person, or the required format and contents of a report. Existence denotes whether the artefact yet exists – is it produced by another department or organisation, or does an activity to create it need to be modelled?

SLIDE 13 (Modified FRAM – Aspects General)

Whilst we have created and defined the use of aspects, occasionally an aspect will not be necessary, and a simple text box will suffice. An example of this is the aspect Time. There is no need to try to model the existence, nor the quality attributes of time, and a simple text box will suffice. For Time, one can simply write ‘by Phase 3’, or ‘by the end of June’ using free-text within the box shown here.

We have also introduced other new features in FRAMSP, as they were needed to cater for the specifics of our research, and these new features are now discussed.

SLIDE 14 (Modified FRAM – Layers of Abstraction)

Some lifecycle models can become large, monolithic, and somewhat unwieldy if left ‘unchecked’. To prevent the creation of large, unusable models we’ve facilitated a form of modularisation by establishing a means of representing different layers of abstraction (such as platform, system, and item). Having allowed for the creation of separate modules within the same model, we have established 2 means of representing ‘off page links’. Such links are used to indicate to the analyst the title of the module which must be read to see the relevant connected activities and artefacts. The first way to represent an off-page link is where the module to be read is obvious by virtue of the name of the activity or artefact. For example, it is clear that the ‘System Architectures’ artefact will be found in the ‘System’ module. In such cases the artefact or activity is shaded entirely grey and the ‘intersect symbol’ is placed after the name of the artefact. There are also artefacts and activities whose location in another module is not clear from its title. In this example, the artefact ‘Interface Requirements’ has the module identifier in italicised, parenthesised text below the title of the artefact. As an off-page link, it is also shaded grey.

SLIDE 15 (Modified FRAM – Artefacts linked by Artefacts)

Artefacts are produced and/or consumed by activities, and the expended resources to produce or consume them are therefore considered by the modelling of activities. As such, artefacts cannot link directly to other artefacts without a producing or consuming activity as a bridge between them. Should a lifecycle under analysis infer a connection between 2 artefacts WITHOUT a connecting activity, then this should be represented by colouring the connecting lines red as shown here.

SLIDE 16 (Modified FRAM – Inferred or Assumed Elements)

Occasionally, a lifecycle does not explicitly consider activities or artefacts. Sometimes activities or activities are merely inferred by a lifecycle (such as a standard stating that “assumptions are vital” without considering their creation and management). Sometimes activities or artefacts are assumed (for example having to assume that “candidate designs” progress to final design specification). Should inferred or assumed activities be apparent, or need to be modelled to achieve internal consistency, then the activity or artefact should be created and shaded in red as shown here.

SLIDE 17 (Modified FRAM – Optionality)

Sometimes it is unclear from reading a process instruction or activity description whether certain activities could or should happen, and multiplicity or optionality are sometimes not explicitly considered by software safety lifecycle processes. In the example shown here, it is unclear whether one or all of the activities should be carried out as a means of identifying system requirements. As such the output aspects are shaded amber as shown here.

SLIDE 18 (Modified FRAM – Optionality)

To solve the issue of optionality and even one for multiplicity, we find that Goal Structuring Notation (or GSN) symbology covers optionality and multiplicity well, and we adopt the same symbology shown on screen in FRAMSP.

SLIDE 19 (Modified FRAM – Referenced Artefacts)

Organisations often have numerous documents which comprise their lifecycle, but not all of them are provided to 3rd-party analysts, and this could be for many different reasons. Open Standards also refer out to external documentary artefacts which do not form part the standard in question – and whilst these may be informative OR normative, they are not provided as part of the Open Standard itself. In both of these cases, these referenced artefacts may be judged by an analyst to be useful or necessary to the successful completion of the activity or activities. Should the analyst be unable to access these artefacts (if IPR, financial cost, or time pressure prohibit it), then such artefacts or activities are shaded black (with the text in white font) to denote that no analysis on their ‘goodness’ has been performed.

The Tutorial is now complete, and you will shortly be asked to undertake a short exercise. Do you have any questions on the use of FRAM to model software safety activities?

SLIDE 19 (Process)

(Start of Practical)

(Questionnaires issued at the end of the practical) – ask for them to be completed and returned immediately whilst the process undertaken is still fresh in their minds.