How System Analysts Can Utilise LLM's to Generate Initial System Models for the FMV

David Slater

Abstract

Large language models are transforming the way we capture and explore complex workflows by automating the creation of FRAM models that until now demanded painstaking manual effort. Instead of laboriously crafting XML by hand, an analyst can simply describe a process in natural language and receive back a ready-to-load .xfmv file for the FRAM Model Visualiser. Under the surface, the model parses the narrative to identify every discrete activity—or "function"—traces how the outcome of each activity feeds into the next, and writes out the corresponding <Function>, <Input>, <Output>, and <Aspect> elements in FMV's XML schema. The result is an interactive diagram of functions and couplings that can be immediately inspected or simulated.

In this article, we delve into this method and walk through two concrete examples—making a cup of tea and preparing a ferry for departure—complete with illustrations drawn from the actual .xfmv files.

Introduction

Large language models (LLMs) are revolutionizing the way we formalize and visualize complex work systems. By encoding the Functional Resonance Analysis Method (FRAM) into a simple code template that utilises the FRAM Model Visualiser's .xfmv XML schema, an LLM can transform a textual process description into a standard FMV model. The prompt for the LLM should first specify how the LLM should perform the FRAM analysis.

The prompt used here :

- describes the system to be studied and asks for an overview of what the system is, how it is supposed to work, what it produces, and any issues known to be a problem or for which improvements have been made. The LLM clarifies which actors, tools, documents, and physical environment constitute the scope.
- Asks for the key functions involved and the tasks they have to fulfil to successfully operate the system, to be listed. Every discrete activity—what FRAM calls a "function"—is listed, each with a unique numeric IDNr, a clear IDName, a FunctionType (foreground, background, variable), and optional display coordinates (x, y) for layout within the Visualiser.
- Invites the LLM to use a hierarchical Task analysis format to determine the sequence and couplings between these functions needed. The LLM performs a lightweight Hierarchical Task Analysis, ordering the functions sequentially and identifying the hand-offs. Each hand-off becomes an Output in the upstream function and a matching Input in the downstream function, sharing the same label (the IDName of the coupling).
- Instructs to use a Python template (from the GitHub website), to build the XML tree.

The final output is a .xfmv file ready for drop-in import to FRAM Model Visualiser, where dynamic playback or FMI interpretation can begin immediately.

Illustration 1: Tea-Making FRAM Model

To illustrate, imagine the quotidian ritual of making a cup of tea. The analyst might describe it as: "First, fill the kettle with water. Next, bring the water to the boil. Meanwhile, place the teabag in the cup. Then pour the boiling water over the bag and let it steep. After steeping, remove the teabag, add milk and sugar as desired, stir, and finally serve." Within seconds, the LLM translates those words into ten numbered functions—"Fill kettle" through "Serve tea"—each positioned in a left-to-right sequence for visual clarity. It infers nine hand-offs, labelling them "Water in kettle," "Boiled water," "Teabag in cup," and so on, and embeds them in the <Outputs>, <Inputs>, and <Aspects> sections of the .xfmv file. When this file is opened in FRAM Model Visualiser, the user sees a clean chain of functions and couplings (see Figure 1), exposing at a glance where variability can enter the process—perhaps the boil takes too long, the teabag placement is imprecise, or stirring is forgotten.

In the tea-making example, ten functions are laid out from left to right—"Fill kettle" through to "Serve tea. The LLM can then generate a make_tea.xfmv file.

When loaded into FMV, this produces a clear, linear network: a node for "Fill kettle" connected by a labelled edge ("Water in kettle") to "Boil water," and so on through the entire tea-making sequence.

Figure 1 – The Tea making FRAM model produced

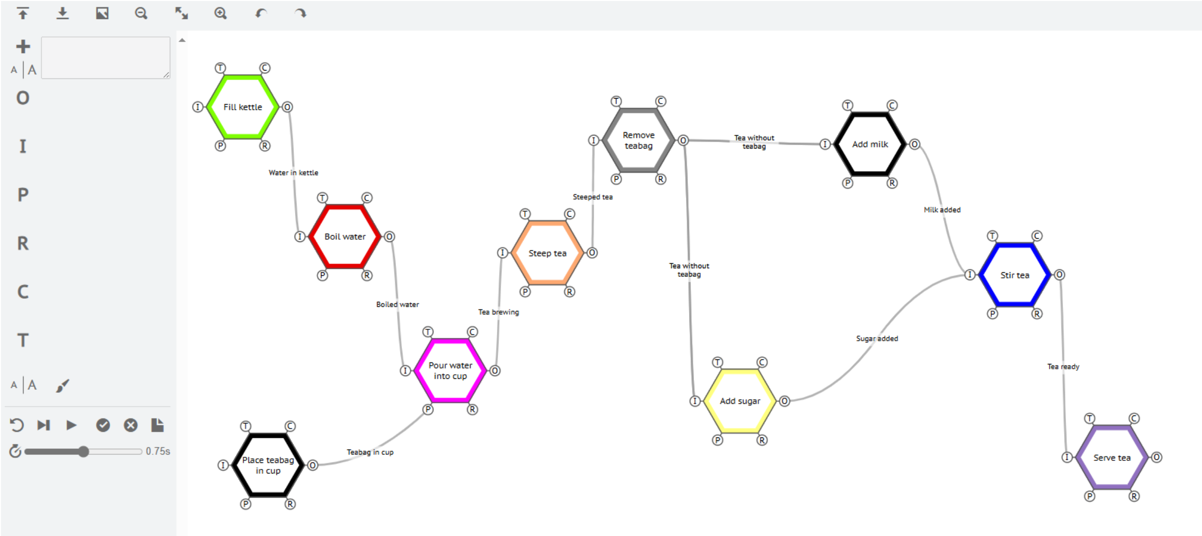

Illustration 2: Ferry-Loading FRAM Model

A far more complex example follows the same recipe. In the ferry-loading scenario, the system boundary begins when cargo loading completes and ends when the vessel departs on its sailing schedule. Between those bookends lie a series of preparatory steps—"Trim vessel," "Man harbour stations," "Drop moorings," "Close bow doors," and "Leave harbour"—supplemented by background functions such as consulting the ship's standing orders and supervising the crew. The LLM maps these eight core activities into an ordered network, creating an output for each step (e.g. "Moorings dropped") and matching it to the next function's input (e.g. "Moorings dropped" as the precondition for "Close bow doors"). The resulting .xfmv file reveals a richly interconnected visualisation (see Figure 2), showing not only the linear departure sequence but also where organizational controls and resources weave in, spotlighting potential bottlenecks or points of failure.

The ferry example scales the same method to eight primary steps, augmented by organizational and background functions. Here is the FMV visualisation of the ferry_model_hta_coupled.xfmv, capturing the core departure sequence:

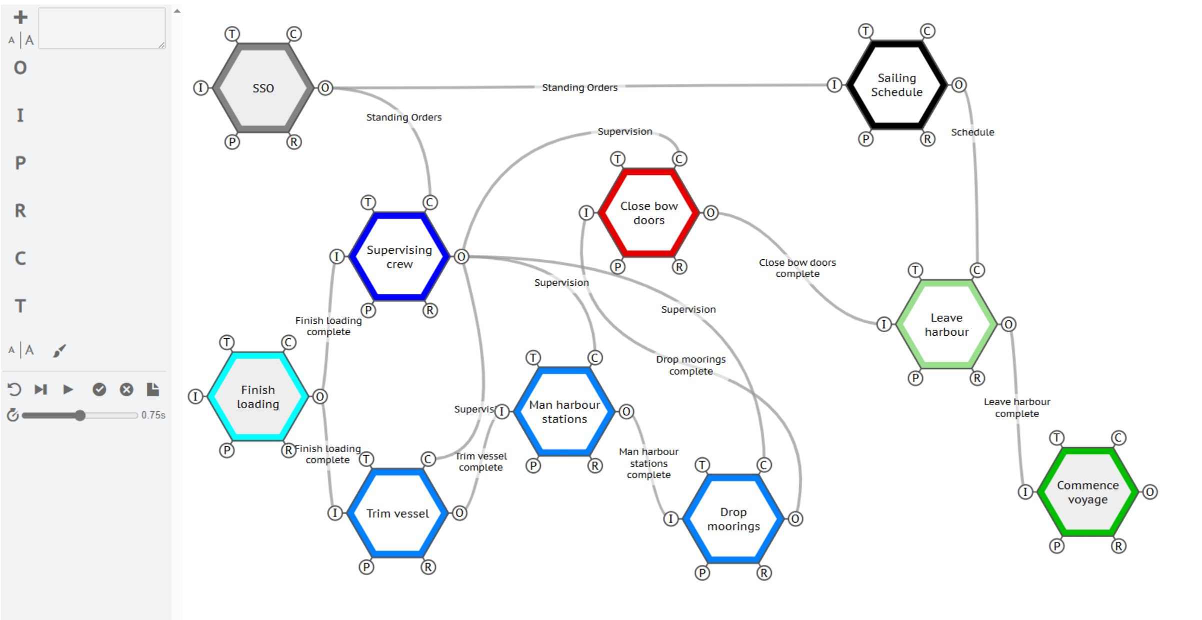

Figure 2 – The Ferry loading FRAM model produced

Visually, FMV will render “Finish loading” feeding into “Trim vessel,” then into “Man harbour stations,” and so on, ending at “Commence voyage.” Background functions like Ship’s Standing Orders and Supervising crew can be added in the same fashion, with their own aspect types (e.g., controls and preconditions).

Conclusions

What sets this approach apart is its seamless blending of conversational expertise and formal modeling. Domain specialists need only speak or write their processes; the LLM undertakes the heavy lifting of assigning identifiers, ensuring consistency of labels, and generating well-formed XML. Analysts can then import the model into FRAM Model Visualiser and immediately begin exploring what-if scenarios, tracing how a delay in one function reverberates through the network, or overlaying metadata for deeper FRAM Model Interpretation analysis. There is no compromise in rigour: the same XML schema that experts once hand-coded is produced automatically, but with far greater speed and far fewer human errors.

By removing the tedium of manual XML editing, LLM-driven FRAM generation empowers analysts to focus on what truly matters—understanding system resilience, uncovering hidden interdependencies, and designing interventions to reduce unwanted variability. Whether it’s the simple joy of a perfectly brewed cup of tea or the high-stakes choreography of a ferry leaving port, the combination of natural-language prompts and automated .xfmv output offers a faster, more accessible path to FRAM modelling than ever before.

Benefits and Outlook

By embedding domain logic into the LLM’s prompt—teaching it the .xfmv schema and providing a reusable code template—analysts can:

- Rapidly prototype initial FRAM models from narrative descriptions.
- Automate tedious XML authoring, reducing syntax errors and freeing practitioners to refine model semantics.
- Maintain consistency across domains, reusing the same pipeline for healthcare, transportation, manufacturing, or software processes.
- Enable iterative refinement: once the skeleton model is loaded in FMV, experts can tweak labels, fine-tune curve placements, or add metadata for advanced FMI analysis.

In essence, this approach marries the LLM’s natural-language understanding with FRAM’s rigorous structure and the FMV’s interactive power. From the quotidian chore of brewing tea to the complex choreography of dock-side ferry operations, an LLM can scaffold the first draft of a FRAM model—transforming human prose into a navigable, explorable visualisation at the click of a button.

Looking ahead, this capability opens doors across industries. In healthcare, clinicians could describe patient-care pathways and instantly generate FRAM models to examine how treatment variations might cascade. In manufacturing, engineers could map production lines to identify where resource shortages or quality controls might introduce variability. Even software deployment pipelines, logistics networks, or emergency response procedures could be captured in a matter of moments. The underlying process remains constant: define the boundary in prose, enumerate the functions, trace the couplings through HTA, and export the .xfmv.

-