The Functional Reasoning Design Language (FRDL): Supporting Hazard Analysis with Graphical Models of Behavioral Interaction

Daniel Hulse[1]
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035
Email: daniel.e.hulse@nasa.gov

Seydou Mbaye
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035
Email: seydou.mbaye@nasa.gov

Lukman Irshad
KBR, Inc.
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035
Email: lukman.irshad@nasa.gov

Functional Hazard Assessment (FHA) is a key early-stage engineering process that supports the incorporation of safety in design by identifying the high-level functional hazards the system may encounter. Functional Hazard Assessment is often supported by models of system functionality that abstract system behavior at a functional level. However, the diagrams used to create these abstractions are limited in what types of behavior they can coherently represent–often leaving out or misrepresenting the key behavioral interactions that cause hazardous conditions to propagate from one function to another. To resolve these limitations, this paper introduces the Functional Reasoning Design Language (FRDL), a formal modelling language for describing the functional elements of a system and their behavioral interactions. To demonstrate the use of FRDL, an autonomous rover system is analyzed using FRDL, showing an improvement in the ability to identify hazard causes and effects when compared to a traditional Failure/Fault Analyses Procedure. This demonstration, along with a qualitative comparison of FRDL with other diagramming conventions used to support FHA, highlights the potential for FRDL to improve the analysis process by more rigorously representing the causal mesh of behavioral interactions that enable hazardous effects to arise from faults or adverse conditions.

Keywords: Hazard Assessment, Functional Modelling, Risk Analysis, Model-Based Systems Engineering

1 Introduction

Functional Hazard Assessment (FHA) is one of the first analyses performed in the safety assessment process and is used to identify the high-level functional hazards that could occur in a system [1]. The FHA process is a key part of the safety assessment and safety-informed design process for two reasons. First, because it is conducted prior to design activity, it can have a significant impact on driving requirements and decisions supporting system safety (which one would expect to have more impact on the overall system development process [2]). Second, it is performed not just at the component level, but for each system and subsystem [1,3], meaning the process will be performed several times in design process as the design becomes more detailed. FHAs are key for establishing an overall safety strategy and can ultimately feed into more detailed analyses of safety such as preliminary system safety analyses (PSSAs) and system safety assessments (SSAs) [4]. However, the FHA process for novel systems can be subject to ambiguities that can make analysis difficult, including a lack of agreement about how to decompose a system into constituent functions.

Given its importance, improving the FHA process has significant potential to the overall effectiveness of the safety-informed design process. While FHA-like "function-level" hazard analyses have been performed from the origin of the safety engineering field (as a part of MIL–P–1629 FMECA processes and their derivatives), the delineation of FHA as its own process is somewhat more recent, first being featured in the aviation safety standard ARP-4761 [1] in 1996 and then in the military safety standard MIL-STD-882E in 2012 [3]. In these standards, the designer identifies the major functions to be accomplished in the system and the hazardous conditions which could affect them. While aerospace standards do not mandate the use of any model or diagram to inform the process, official guidance specifies that the analyst creates a hierarchical functional decomposition that represents composition relationships between high-level functions and their component sub-functions and sub-sub-functions (for examples, see: [5,6]). Other standards, such as the ARP-926C [7] Fault/Failure Analysis (F/FA) Procedure defines specific input/output diagrams for high level "top down" functional analysis of hazards and flow chart-style diagrams called "functional flow block diagrams" for "bottom up" hazard analysis. Block diagrams like this are thought to be more supportive of hazard analysis of highly integrated systems [8], but are subject to limitations in their ability to abstract the true behavior of the system. Additionally, to better accommodate the hazard analysis of autonomous and more-autonomous systems, automotive standards now suggest the use of control block diagrams (per the use of the STAMP/STPA [9] methodology) for analysis of functional hazards [10].

Over the past 25 years, two major lines of research have proposed various improvements to the system representations used to support early design hazard analysis. In the engineering design literature, much has been done to tie the idea of functional failures (identified in FHA) to functional modelling frameworks (i.e., energy-materials-signals models described by Pahl and Beitz [11]) proposed by design theory [12,13]. Particularly, models in which the functions of the system act on flows of energy, material, and signals have been demonstrated in the context of identifying faults and their propagated effects through the system [14–16]. In the field of system safety, on the other hand, there has been much activity applying the ideas of system science to the understanding of how hazards arise [17,18]. One of the most influential methodologies has been Systems Theoretic Process Analysis (STPA) [19], which is based on the Systems Theoretic Accident Model and Pro-

[1]Corresponding Author.
Version 1.26, June 26, 2025

cess (STAMP) hazard model that considers the feedback structure between processes and their controllers (e.g., software, operators, etc) [20]. STPA frameworks have seen significant interest from industry because of their ability to represent and analyze accidents by pushing the scope of hazard analysis from the designed system (i.e., inputs and outputs of a function) to the high-level interactions between the system and its operators.

While these research efforts have provided important theory for improving the FHA process, they are often difficult to extend beyond their narrow scope of research interest (e.g., modelling mechanical systems, socio-technical interactions, etc.) to more general applications. In previous work, the authors have developed simulation tools for modelling hazards arising from operators, physical/mechanical behaviors, as well as their interactions [21,22]. As a practical necessity, this has led to the development of *generic, abstract modelling structures* for representing the functions of a system and their interactions in a way that is amenable to behavioral simulation. This paper applies the insights from this development work to motivate and develop the Functional Reasoning Design Language (FRDL), a graphical language for representing functional architectures in the FHA process. The aim of FRDL is to rationalize the development of FHA by enabling the representation of behavioral interactions between functions to enable a more rigorous analysis of fault and failure propagation. The major contributions of this work are thus (1) the identification of issues present in current early design representation used (in practice and in literature) for FHA, (2) the identification of principles that functional modelling languages should embody to address these issues, and (3) the introduction of the FRDL language which embodies these principles.

The organization of this paper is as follows. First, some basic background on Functional Hazard Assessment and related research is presented in Section 2. Then, the issues with these approaches, definition of principles, and proposed FRDL modelling constructs will be presented in Section 3. Next, FRDL will be demonstrated in Section 4 and compared qualitatively against other FHA-supporting diagrams in Section 5. Finally, conclusions will be presented in Section 6, along with lessons and avenues for further work.

2 Background

This section discusses the state of standards and ongoing research in FHA to motivate and contextualize the development of FRDL in Section 3.

2.1 FHA Standards. The FHA process is defined in the military standard MIL-STD-882E [3], as well as the aerospace standard ARP-4761 [1] and FAA Advisory Curricular AC.23.1309-1E [23], as a process supporting the identification of product-level hazards which may occur, as well as their high-level effects.

Generally, these standards do not specify the process of performing the analysis in detail (though there may be supplementary guidance and appendices providing examples of the process) as much as the format and meaning of the output (e.g., tables, standard severity classifications, etc). However, the analysis process for performing FHA is similar to other well-known discursive hazard analysis approaches such as Failure Modes and Effects Analysis and Hazard and Risk Analysis (defined in the ISO-26262 "functional safety" standard for automotive industries [24]), except that it doesn't include an assessment of rate or probability, since that information is not available at the function level.

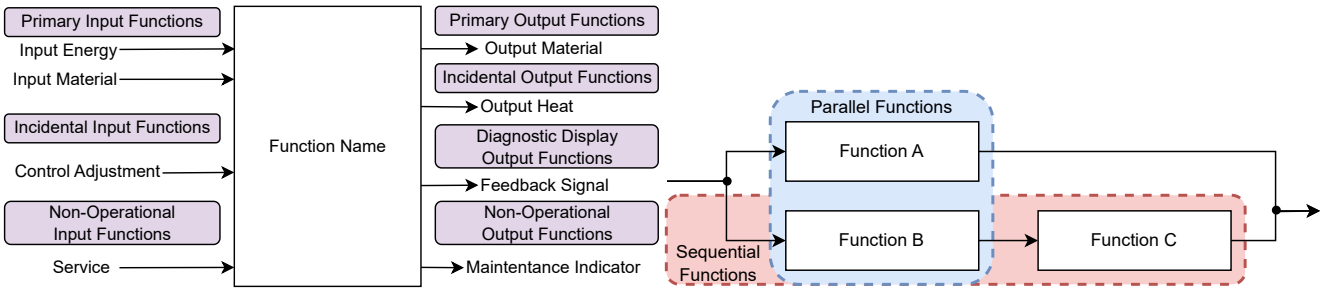
While standards vary by industry, the generic, broadly-applicable analysis aspects of FHA may be gleaned from the Fault/Failure Analysis (F/FA) procedure described in ARP-926C [7], which descends from the early military standard for FMEA (MIL–P–1629) and generalizes the procedure in a way that can be applied both at the function and component level. For function-level analysis, ARP-926C describes a "Functional Approach" to F/FA, which may be performed using "top-down" or "bottom up," analyses. In the "top-down" approach, functional analysis is performed in a manner that views the overall function of the system as a single block, with its input functions on the left of the block and output functions on the right of the block, as exemplified in Figure 1(*a*). This is used to determine the hazardous conditions (failure modes, poor inputs, etc.) and their effects in the context of the overall system function (i.e., inputs and outputs from the function). The other approach, low-level "bottom-up" analysis, creates a functional flow block diagram of the system as shown in Figure 1(*b*). In this type of diagram, the overall function is split into a set functions that interact via arrows, where forks in arrows can be used to represent functions performed in parallel with each other, rather than in sequence. This diagram essentially represents the overall function of a system as a "task" that is composed of sub-tasks that, when completed, perform the overall task.

Standard diagramming conventions for function-level analyses can differ substantially when considering software and human factors. Notably, for digital systems and equipment (ARP-1834 [25]), functional block diagrams show the interaction of electricity and/or signals between functions. In this domain, signal connections can be specified using *bidirectional arrows*, since communications may go back and forth between the individual functions or components using the same channel. This is different from a functional flow block diagram, where the arrows only flow in one direction through the system. Additionally, the type of errors considered by software failure analysis are often quite different than physical systems, since faults are expected to arise from development errors rather than mechanical processes [26]. Recently, in the autonomous vehicle domain, the ISO-21448 standard has introduced more methods to assist with hazards related to autonomous and partially-autonomous systems, including the use of systems-theoretic models to examine hazards arising from the interactions between the driver, the vehicle, and the environment [10] such as STAMP/STPA [9]. This has been motivated by an increasing interest in accounting for complex interactions that happen at the system/vehicle level that can lead to failures (e.g., software that causes a driver to crash).

To summarize, while FHA-related standards agree in the general approach of identifying the system functions, hazards, and resulting effects, the underlying models used to analyze the system in support of this are not universally agreed upon across industries and can vary by domain, especially as FHA is applied to novel autonomous systems. The aim of this paper is to provide a single FHA-supporting language which can be used consistently across domains and industries while accounting for the complex interactions present in autonomous and highly-integrated systems that are currently challenging the state of practice.

2.2 FHA Literature. Literature on FHA can be broken down into examples of FHA to novel use-cases (such as aeroelastic wings [27], virtual control towers [28], software [29], and AI/ML functionality [30]) and the development of methodology to support the FHA process. While performing an FHA only requires a list of system functions to identify hazards and their effects, it is considered good practice to inform the process using function-flow block diagrams (see Figure 1(*b*)) [31], especially when designing highly-integrated systems [8]. Methodologies for FHA in the literature have thus provided ways to improve the underlying language used to represent functionality, by applying defined modelling languages and/or simulation techniques to inform or perform the analysis.

One of the most common conventions for representing function in the FHA literature is with a hierarchical containment model. In this convention, primary functions of the system are decomposed into sub-functions and organized in a hierarchical tree until an acceptable level abstraction is achieved [5,32]. One of the obvious failings of this representation is that it doesn't capture the behavioral interactions between functions, and is thus no different than a hierarchical list. As such, methodological extensions to this approach have enabled the representation of relationships between functions. For example, the Goal Tree Success Tree Master Plant Logic Diagram (GTST-MPLD) provides a means to con-



(*a*) Template Function Diagram (*b*) Template Function Block Diagram

Figure 1 Diagram types used in the Function/Fault Analysis Procedure

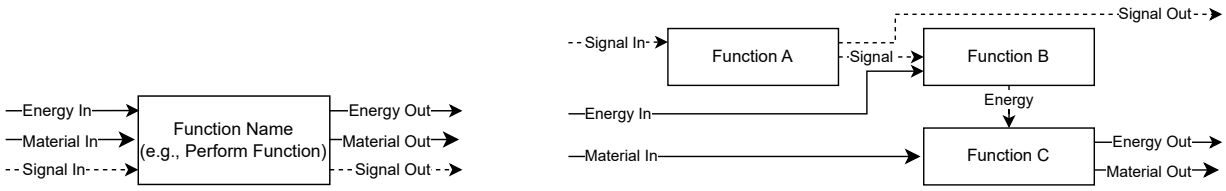
sider interactions across a model hierarchy. This is provided by the use of "AND/OR" gates in the context of an overall function/component hierarchy, as well as providing dependency matrices to track interactions between functions [33]. However, this approach requires much more detailed analysis across multiple diagrams and thus loses some of the advantage of a functional perspective. Other frameworks augment the hierarchical model with separate (functional-block-diagram-like) diagrams which include "input/output" and "method/constraint" arrows [34]. While these methods enable some ability to assess interactions, they add substantial diagram complexity without fully representing the means by which functions interact–instead representing the interactions as arrows. Thus, while these methods improve function-flow block diagrams, they still are limited in their ability to represent the behavioral interactions that ultimately enable failure propagation in hazard analysis.

In the engineering design field, there has been much interest in using formal functional modelling languages to support FHA-like analyses. Particularly, energy/materials/signals (EMS) models such as the Functional Basis Engineering Design (FBED) [35] and others [11] have a similar structure to function-flow block diagrams, as shown in their respective templates in Figures 1 and 2. The main difference is that when developing a functional decomposition from a high-level functional model, they apply a "spacio-temporal" representation that ties the sequence of functions to their input-output relationships, as illustrated in Figure 2(*b*), enabling a better understanding of the physical interactions between functions. Using these models has been shown to assist with the generation of FMEA-style hazard tables [36], and the underlying modelling language has been extended in the context of hazard analysis to enable the representation of human actions and errors [37–39]. However, as with Function Block Diagrams used in the Function/Fault Analysis Procedure, EMS-based modelling approaches are somewhat deceptive in the context of failure analysis because they represent the system as a directed graph, which makes it difficult to represent bi-directionally interacting behaviors and important feedback loops. These aspects are important for analyzing key interactions between the system, the environment, and human operator(s), since these are generally considered to be outside the system boundary.

Modelling and simulation has been an active area of interest for formalizing, informing, and iteratively developing hazard analyses. Many of simulation approaches have been developed using EMS-type functional models [13]. In this area, early methods for simulating hazards in functional models involved creating component models of the system to propagate to the functional level [14]. These models have further been augmented with action sequence graphs to simulate function failures and human error propagation in tandem to support joint errors [40]. A key insight from simulation has been that functions often have inherent associated behaviors which can be modelled without fully specifying components [15], as illustrated in Figure 5. That is, functions represent defined physical behaviors which may be realistically modelled without fully specifying components. This has led to modelling frameworks that represent the behavior of a system primarily as interactions between functions [21], which in turn have been augmented to further represent human behaviors and hazard by including human-oriented modelling constructs such as action sequence graphs, performance shaping factors, and information networks [22,41].

Outside of the engineering design literature, SysML [42] has been used to inform FHA [43]. While this has been helpful for enabling a model-based hazard analysis paradigm, SysML has no concept of "function" and one is often left using diagrams (e.g., activity or block diagrams) that may not fully represent the structural and behavioral properties needed to represent failure propagation on a single diagram. In the past, this problem has been addressed to some extent by creating an interacting simulation that uses an EMS-based modelling paradigm [44]. Similarly, state charts have been used to simulate behavioral models to inform FHA [45], which is helpful for formalizing behavior but not representing structure, making it difficult to derive failure propagation without modelling it explicitly via states.

More recently, systems theory-based hazard analysis approaches like the Function Resonance Analysis Method (FRAM) and System Theoretic Process Analysis (STPA) have gained attention due to their ability to represent complex interactions between different system elements [46]. STPA [47] uses a control structure where the system is considered a collection of interacting control loops where controllers, controlling processes, and support systems are captured through blocks and interacting feedback arrows, as shown in Figure 3. This enables the assessment of poor control actions–the archetypal cause of "accident"–type catastrophes which have increasing impact as systems increase in complexity. FRAM [17,48], on the other hand, uses a functional representation where a function is described using six characteristics; inputs, outputs, preconditions, resources, times, and control. The interconnection between functions are represented through the connections between function characteristics (e.g., output of one function is a resource to another) and hazards are considered to emerge due to the variability of these interactions. Because of its representation of time-based dynamics and event sequence, FRAM is somewhat more flexible for understanding the dynamic behavior of systems necessary for understanding resilience [49]. Both STPA and FRAM resolve a major problem in EMS-based models, in that they enable the representation of high-impact hazards that arise from the operator and external environment that would be considered to be outside the system boundary (and thus, out of scope of the analysis). However, one major limitation of these methods is that they have not been developed to analyze the (non-control-based) physical/technical aspects of system behavior with as much detail as EMS-based methods. Nevertheless, these methods have seen increasing interest from industry, with STPA being suggested in ISO-21448 as a way to analyze hazard relating to autonomous and semi-autonomous driving [10].



(*a*) Template high-level functional model

(*b*) Template functional decomposition

Figure 2 EMS Functional Models Function Failure Identification and Propagation (FFIP)-type Methods

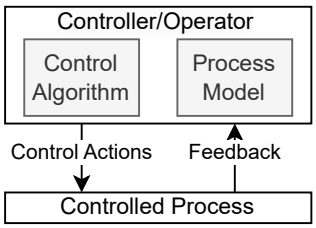


Figure 3 Template STAMP model used in STPA

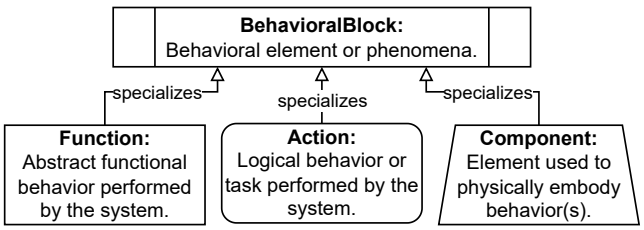


Figure 4 Types of blocks representing system structure.

3 Defining the FRDL

The development of the functional reasoning design language is motivated by the inherent issues present in existing diagrams used to inform the FHA. This section highlights some of these issues, identifies desired principles for an FHA-supporting diagram to fulfill, and uses these principles to justify the modelling constructs which make up the FRDL.

3.1 Defining Behavioral Blocks. One core feature of FHA-supporting diagrams is the representation of the overall system and its decomposition into individual elements. From there, these blocks may be arranged in an overall diagram, such as a temporal block diagram (in ARP-926C [50]), function-flow block diagram (in FFDM [12]), or control structure (in STPA [19]). In general, these approaches agree that a function should be a "form-invariant" representation of the overall "purpose" or "task" performed by the system–a feature they share with EMS-based functional models. However, defining these functions is often an issue for the FHA process, since there is always some difficulty abstracting what is known about the system into functions [8]. While much of this is because the concept of a function is inherently more abstract than known components, it also is a result of Problem 1.

Problem 1: *Existing FHA models impose functional abstractions which may not be fully coherent.* Specifically, there is confusion between the system being made up of "technical functions," which are *functionalities* of the system (i.e., the framework used in the FFDM literature), "tasks," which are discrete modes or events the system performs (a framework implied by function block diagrams), and components, which are the physical elements of the system (a framework which is often used by default when engineers do not readily conceptualize the functional abstraction). Often, rules imposed by diagramming approaches to enforce a single type of abstraction (e.g., a function must be a noun-verb pair acting on inputs to produce outputs, etc.) do not appreciably resolve the ambiguities that create this distinction, leading to confusing models that may have multiple interpretations. This is especially relevant to the design of aircraft, where the overall function of the aircraft is not readily expressed in terms of inputs and outputs and utilized functionality may change over the phases of operation (it should be noted that hierarchical functional decompositions–not block diagrams–are recommended in guidance for hazard analysis in aviation [1], which may be related to this).

Principle 1: *Graphical FHA-supporting diagrams should delineate the different temporal and structural aspects of system behavior.* Specifically, behavioral elements which are considered more as "tasks" have a temporal limitations (e.g., a state in which the task is "complete" or "failed"), whereas behaviors that embody as "functionality" are existent throughout the operation of the system. For example, an aircraft may turn left and right multiple times to accomplish a given mission, tasks which rely on the existence of pitching and general aviation functionality embodied by wing, aileron, and control components. The distinction between these types of behavior should thus be clearly delineated so that they are not confused. An example of this would be how the SysML language differentiates structural and behavioral elements of the system as "blocks" and "activities," respectively, creating a conceptual distinction which avoids confusion. This principle is embodied in Solution 1.

Solution 1: *In FRDL, functions are represented as a type of behavioral element which may contain actions or components.* Functions, components, and actions are abstract behavioral elements of the system, meaning that they *embody phenomena*, as shown in Figure 4. Examples of these phenomena include physical equations (e.g., equations of motion, force balance, etc.), logical operations, and tasks (e.g., pressing a button or taking an object to a location). Behavioral blocks are differentiated by the type of behavior and structure they represent. In this formalism, functions thus represent *abstract functionality* which itself may be further embodied by actions or components. Actions in turn represent the *logical behavior* or sequential tasks performed by the system over the course of an operation, such as taking off or landing an aircraft. Components, on the other hand, represent the hardware elements which will physically embody the system. The relationship between component and function is illustrated in Figure 5 in the case of wheels in a car, where the functions "affect forward accelleration", "support frame," and "control path of travel" are composed of the aggregated individual behaviors from each of the wheels. This behavioral perspective clarifies the role of functions compared to components while imposing a level of formality on what a function can be. Specifically, functions are concrete system behaviors (e.g., movement, acceleration, force balance, etc.) which will be in the system regardless of the specific component architecture. However, while delineating behavioral block types enables a more expressive language for representing system behavior, more information may further inform analysis, which motivates Problem 2.

Problem 2: *Existing FHA-supporting diagrams do not convey relevant structural and behavioral aspects of function behavior.* Generally, FHA-supporting diagrams represent functions solely as boxes with a name for the function, with no further indication

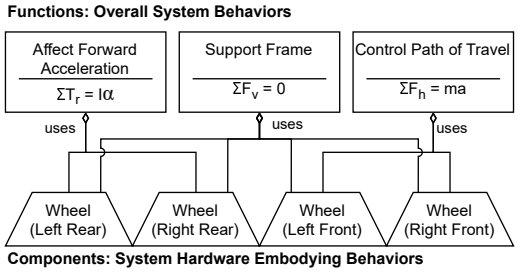


Figure 5 Relationship between functions (high-level behaviors) and components in an automotive wheelbase.

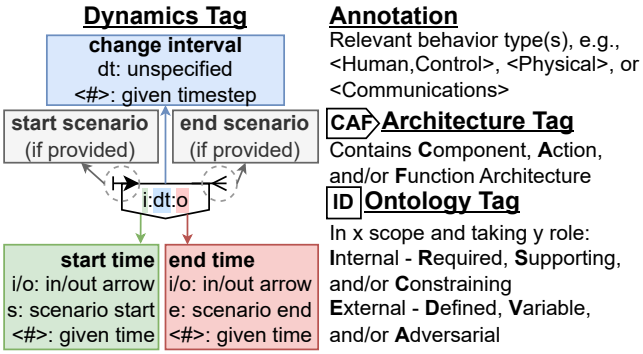


Figure 6 Proposed function tags and annotations

of known characteristics of the function. One exception to this is STPA, which specifies one box as a "controlled process" and another as a "controller," which in turn delineates the type of behavior of each box (as controlling versus controlled behavior) and informs the identification of hazardous behavior associated with each function [19]. Given the other aspects of functions (timing, embodiment, and control structure), solely specifying functions as boxes represents a missed opportunity to use these properties to inform hazard analysis, justifying Principle 2.

Principle 2: *FHA-supporting diagrams should include the flexibility to convey structural and behavioral attributes to inform analysis.* Relevant structural and behavioral attributes should be able to be expressed when representing functions to better inform hazard identification. Properties like timing, embodied components, and parameters can all improve the understanding of hazardous behavior. Thus, to inform analysis, designers should be given a means to convey this information–not as a requirement of the FHA process, but as an option to provide more detail. Providing this capability would further help address Problem 1 by giving designers a means to not lose relevant information about known behavior when applying the functional abstraction. This motivates Solution 2.

Solution 2: *In FRDL, behavioral blocks may be annotated to convey relevant behavioral and structural details.* In order to specify hazard-effecting details which may inform analysis, FRDL enables the use of tags and annotations. These are shown in Figure 6 and explained in the next sub-sections.

3.1.1 Dynamics Tag. An important aspect of functionality for understanding failure propagation is the dynamics of the underlying functional behavior. Behavioral dynamics determine important hazard-effecting properties like fault opportunity (i.e., in which phases of operation a fault may arise), timing errors (highlighted in STPA [19]), and resulting effects (as explored in [51]). This time-related information is important for understanding the effects of hazards as they propagate through the system. To specify this dynamic information, FRDL uses the "Dynamics" tags shown in Figure 6, which state whether the function or action activates at a given time, updates continuously over a given timestep, and deactivates at a particular time. Note that these specifics may not be known early in the FHA process and thus are not required to be specified in full, and should be considered notional, abstract assumptions used to support an analysis, rather than concrete system requirements. For example, tagging a function called "accumulate water" with a timestep of 1 minute means that the task of accumulating water progresses in the scale of minute, not that it must literally accumulate the water every minute. Further means of specifying the dynamics of function behavior in the context of its interactions are defined in Section 3.2.2.

3.1.2 Behavior Type Annotation. As mentioned previously in Section 2, different types of system behavior may have different types of hazards associated with them. In particular, physical behaviors can be modified by physical conditions, and thus have a variety of physical mechanics associated with them that can cause them to fail. Logical behaviors (i.e., control logic), on the other hand, cannot be modified, except insofar as they are embodied by physical processes (e.g., memory, bit-flips, etc) or implementation (design errors). Various types of information about behavior can thus be used to inform the analysis, as exemplified in the literature (e.g., human versus hardware behavior type, see: [19,22]). FRDL represents this behavioral information using the behavior type annotation shown in Figure 6, which allows the designer to represent hazard-relevant type characteristics, such as physical or logical behaviors; high-level tasks like perception, planning, control or communications; or embodying elements like humans, computation, or mechanical elements.

3.1.3 Architecture Tag. One key feature of the functional abstraction is that functions may be broken down into further subfunctions, or, as specified in Solution 1, embodied by components or actions. As this is performed, the overall function may thus be considered to contain architectural information. If these details have been represented somewhere (e.g., on a lower-level diagram), it may be important to convey that information so they may be referenced in the overall hazard analysis. In turn, it should be possible to distinguish functions which have been broken down in detail like that, as opposed to being treated at a high-level (a function to design). In FRDL, this may be represented using the "Architecture" tags shown in Figure 6, where a "C" represents a component architecture, an "A" represents an action architecture, and an "F" represents a functional architecture (see Section 3.3 for definitions).

3.1.4 Ontology Tag. With the increasing autonomous operations of complex engineered system in safety-critical environments (e.g., autonomous vehicles), it is becoming increasingly important to extend the scope of hazard analysis to consider interactions between the system and the environment. However, conveying this from a functional perspective may be confusing–typically, there is guidance to think of functions as intended functionality or "tasks" the system must perform [11]. This is further complicated by the fact that many behaviors embodied by the system or expected by a system in hazard analysis may not be desired [16]. To enable the rational understanding of scope (functions within or outside the system boundary) and role (intended or not), the ontology tag may be used, which provides categories for each, as shown in Figure 6. "I - Internal" functions are those considered within the system boundary and can take "R - Required" (necessary to fulfill functional requirements from stakeholder(s)), "S - Supporting" (needed to support the implementation of required functions, e.g., by providing input or accepting output EMS flows) and "C - Constraining" (expected to exist within the system as a result of physical, manufacturing, or other inherent constraints on system behavior) roles. "E - External" functions are those considered outside the system boundary and can take "D - Defined" (narrowly specified via interfaces, e.g., power supply), "V - variable" (expected to change within or beyond certain bounds, e.g., human operators), and "A - Adversarial" (unstable behavior to be controlled for by the system, e.g., traffic or weather conditions) roles.

These tags provide the ability to understand whether the function is in (or outside) the system as well as *why* the function is included in the model (e.g., to embody a user requirement or to represent a technical constraint or outside environment).

3.2 Defining Flows. In FHA-supporting diagrams that convey function interactions, arrows may represent a few distinct but related properties. In function-flow block diagrams, flow arrows represent sequence, while in EMS-based functional models, flow arrows represent "spacio-temporal" information–meaning, the transference of energy, material, and signal and the sequence implied by that transference [11,35]. This presents ambiguities and limitations that make it difficult to represent bi-directional interactions which unfold over time, as stated in Problem 3.

Problem 3: *FHA-supporting diagrams often apply ambiguous definitions of functional flow which conflate "what" with "when" and "how".* Specifically, arrows can represent mechanics of causality, but can also represent means of interaction, such as input-output relationships of energy, materials, and signals. These arrows confuse the analyst's understanding of system failure propagation, because functional interactions are often bi-directional, meaning the functional failures may not solely impact "downstream" functions later in the sequence of arrows, but also "upstream" functions. For example, a short in a light bulb not only affects the optical energy output it provides, but also affects power input provided to it from electricity sources. This potential for bidirectional causality is not represented in many current FHA-supporting diagrams in part because they encourage the conflation of spacio-temporal order with causality (and thus fault propagation), motivating Principle 3.

Principle 3: *FHA-supporting diagrams should delineate between causal means and mechanics by which functions interact.* It is important to represent both connections (i.e., the "what"–energy, materials, signals and/or shared variables or properties) and causality (i.e., the "how" specifying what properties cause a change in behavior) relationships between functions. Causality is important for understanding how a change in one block may lead to changes in other (e.g., for advancing to the next step in a sequence of tasks). Connections, on the other hand, are important for representing the mechanics by which a change in one function's behavior may effect other functions. Both sets of information should be included to support analysis, but in a way that does not confuse the concepts. This principle is embodied by Solution 3.

Solution 3: *Causality and connections may represented via flow nodes and connection, activation, and propagation relationships.* Flows in FRDL are defined as nodes which connect behavioral blocks in an overall architecture. Flows represent the shared variables that connect functions. Flows may be composed of energy, materials, signals, or other shared aspects (e.g., components, objects, aggregations of properties, etc.). As nodes (as opposed to edges), flows can belong to more than two behavioral blocks, which represents an advancement over traditional representations of function (e.g., EMS models). This enables the representation of more complex interactions between functions typical of complex systems, such as variable coupling (e.g., multiple functions in an aircraft sharing the same position, velocity, attitude, etc). The notion of perception and communication relationships is further expressed by delineating types of flows, as shown in Figure 7, which follows the definitions described in previous work [41] for representing information networks in the context of modelling distributed situational awareness. Here, MultiFlows represent flows which contain multiple copies (e.g., ones perceived by individual functions), while CommsFlows may represent an entire flow network enabling communications (i.e., flow copies communicated between different functions). These containment arrows and flow types provide a rich language for understanding how shared properties flow between functions.

Three main types of relationships are defined to related behavioral blocks and flows, as shown in Figure 8, which represent the connection of the block to the flow, how the flow activates blocks (and vice versa), and how the propagation of flow characteristics

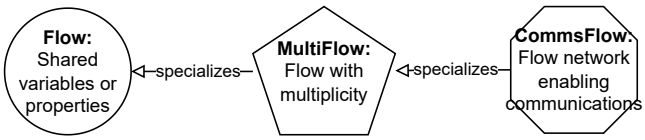


Figure 7 Types of flows

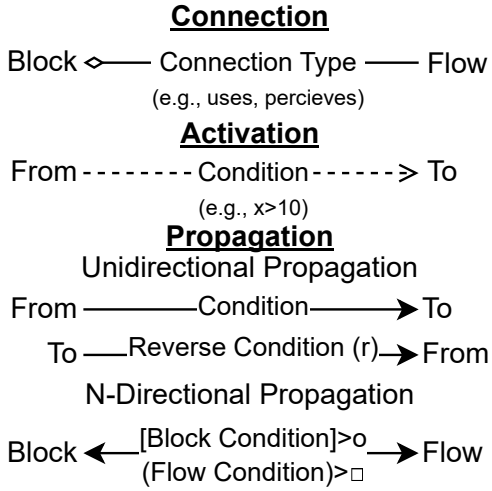


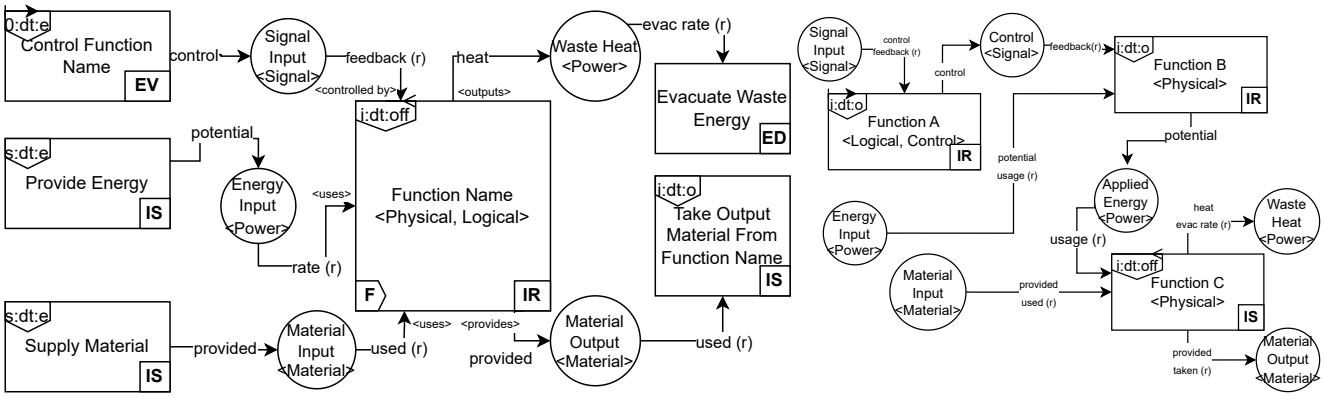
Figure 8 Options for representing flow containment and propagation relationships.

between blocks. These are described in detail in the following subsections:

3.2.1 Connection. With flows defined as variables and properties shared between behavioral blocks, connections (a relationship similar to the aggregation relationship in SysML) may be used to describe whether a flow belongs to a given block. These connections are represented with a diamond at the block end and a line at the flow end, as shown in Figure 8. Connection type annotations may additionally be used to give more insight into how tightly the flow couples the blocks it connects. For example, an "uses" annotation could be used to convey that the block takes the flow as input and modifies it (which may be a strong coupling), while a "perceives" annotation could be used to convey that the block copies the input (which may be a weaker coupling).

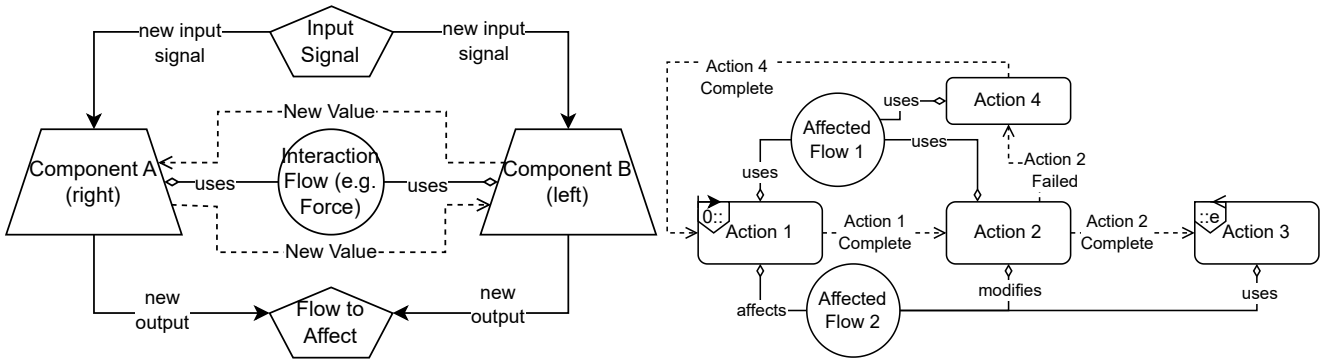
3.2.2 Activation. In FRDL, the activation behavior may be specified separately from sharing of flows to enable a clearer picture of the interactions between functions. Activation refers to how the behavior in one function (or values of a flow) changes or updates the behavior of a connected function. This idea originates from the simulation of tightly-coupled behavioral models, where simulating one functional block often requires re-simulating connected function to propagate failures and achieve consistent behavior. In previous work, this has been accommodated via "static propagation" algorithms in fmdtools functional models [21] and conditions in action sequence graphs [22]. Outside of simulation, conveying propagation has potential to rationalize hazard analysis process by making the tabulation of effects from initiating causes more legible and explicit. To achieve this, FRDL provides the activation arrows shown in Figure 8, which may be used to describe how a condition from one block will result in changed behavior in another block via their flow connections.

3.2.3 Propagation. Finally, propagation arrows, shown in Figure 8 represent both the connection of blocks and flows, as well as the activation or update behavior that is carried by the flow. Propagation arrows are represented with an arrow along with the



(*a*) Function-In-Context Diagram for Figure 2(*a*)

(*b*) Functional Architecture Corresponding to Figure 2(*b*)



(*c*) Template Component Architecture Diagram

(*d*) Template Action Architecture Diagram

Figure 9 FRDL architecture types supporting functional, component, and controls/task-oriented hazard analysis.

activation condition(s) it carries. Propagation arrows may be unidirectional or n-directional, and can optionally be annotated with type tags (e.g., <uses>) at the end of the edge corresponding to the behavioral block, as shown in Figure 9(*a*). Unidirectional arrows convey a singly reversible direction of propagation, with activation conditions propagating along the direction specified by the arrow and reverse activation conditions (marked with (r)) propagating in the reverse direction.

N-directional propagation arrows carry multiple activating conditions between blocks and flows using the notation shown in Figure 8, where [condition]> ∘ represents a block condition activating a flow and (condition)> □ represents a flow condition activating a block. While propagation arrows convey the same information as activation and connection arrows, they are recommended for use when flows connect more than two blocks or when a block is expected to be decomposed in a lower-level architecture diagram, since they concisely group multiple related flow properties.

3.3 Architectures. FHA-supporting diagrams can be thought of as graphs that relate nodes (blocks) with edges (connections, flows, or other relationships) to represent the overall structure of system functionality. This graph can then be used to help trace the propagation of hazardous conditions from one function to other functions. As described in Problem 3, many existing FHA-supporting diagrams are limited by the information represented by edges in the context of this graph. However, they are also held back by the limitations of the graph structure, as stated in Problem 4.

Problem 4: *FHA-supporting diagrams inadequately represent means of interaction (e.g., flows) that connect more than two functions.* Specifically, the since these diagrams represent flows as edges that connect nodes (that themselves represent functions), it is only possible represent the propagation of flows between two functions. This makes it difficult to represent multi-function interactions that occur between more than two functions, such as multiple functions interfacing with the same environment, multiple functions communicating with each other as a part of a network, or functions having highly coupled aggregate physical interactions (e.g., force balance or heat transfer). To represent these n-function interactions in a diagram where flows are represented as edges, multiple flows (copies representing the same instance) must be used to connect the functions of the graph. For example, a robot may have multiple functions, such perception, manipulation, and locomotion that interact with the environment. Representing the interactions between these functions via an environmental flow would require placing three edges labelled "environment" to connect all of the functions. This is problematic for a few reasons: (1) it creates complexity in the diagram as well as the analysis process, since shared interactions between n functions require $\frac{n*(n-1)}{2}$ flow edges which must then must be traced during hazard analysis, (2) it introduces ambiguity in the process of interpreting diagrams because it is difficult to differentiate multiple flow edges representing the same flow (e.g., a shared environment) and flow edges representing copies of the same flow type (e.g., multiple flows of electrical power), and (3) it creates opportunities for missing required shared connections in the diagram (e.g., if A and B are connected by C and A' and B' are connected by C, the analyst may not realized that A, B, A', and B' should all be connected via C). This justifies Principle 4.

Principle 4: *FHA-supporting diagrams should tractably represent flows present in more than two functions to support the analysis of propagation.*. If an FHA-supporting diagram is meant to show the means by which functions interact in order to aid causal analysis, it needs to readily elicit all potential propagation pathways. In order to do so, flows should be able to be connect more than one function while being recognizably distinct (e.g., the same flow should not be present on the same diagram unless it is an independent copy). This is realized in Solution 4.

Solution 4: *System architectures are represented with bipartite*

graphs of blocks and flows, where edges may represent containment and propagation relationships.

In FRDL, architectures are represented using bipartite graphs of functions and flows to better express behavioral interactions between functions. Bipartite graphs are used in graph theory to represent the relationships between one set of nodes via another set of nodes [52], a representation that can be helpful for mapping how entities relate (rather than simply mapping that they relate). For example, bipartite graphs have been used to analyze how different diseases (one set of nodes) share genes (another set of nodes) to better understand the similarity of these diseases [53]. In FRDL, a bipartite representation enables the connection of more than two functions with the same flows, since flows are represented as nodes, rather than edges. Edges in this type of graph in turn represent the relationships (e.g., connection, propagation, etc) between functions and flows. This provides a more tractable representation of coupled system interactions that scales better as more interactions are added, since flow objects do not need to be duplicated and fewer edges need to be added (1 instead of n) as new functions are connected via flows, resolving Problem 4. A template for the bipartite representation used in FRDL is shown in Figure 9(*b*). In this representation, flows may be shared by multiple functions without inherently conveying an input-output relationship. Instead, propagation information is specified explicitly via propagation arrows. This is an important consideration for modelling highly-coupled physical systems, where different properties flow through the system in different directions. For example, in a valve system, water may flow in a defined spacio-temporal direction, but a blockage of water may cause a backup of pressure, leading to faults considered "previous" in the functional flow. This overall system representation may be specialized by block type to further describe different types of system architectures.

3.3.1 Architecture Types. Considering the three types of behavioral blocks presented in Section 3.1, three main types of system architecture diagrams may be defined–functional architectures, action architectures, and component architectures.

Functional Architecture Diagrams show the functions of the system and their interactions via flows and propagation/activation arrows. These diagrams are important for conveying the high-level behaviors of the system and their interactions with each other, as illustrated in Figure 9(*a*) and Figure 9(*b*). One common characteristic of function architecture diagrams is that interactions between functions may be based both on updated information (e.g., if a flow changes in one function, it changes another connected function) as well as the unfolding of behavior over time. As a result, it can be useful to use the time-based behavior tags (in addition to propagation arrows) to specify how these functions may be activated.

Two major functional architecture diagrams may be used to support the FHA process. In the top-down analysis of hazards, a **Function-In-Context Diagram**, which shows the overall functionality provided by the system and its interactions with external functions (see Figure 9(*a*)) may be developed to support the process. In the bottom-up analysis of hazards, on the other hand, the **Functional Architecture Diagram** may be used to analyze the individual functions of the system and their interactions with each other (see Figure 9(*b*)).

When analyzing the logical behavior of the system, **Action Architecture Diagrams** may be constructed to convey the sequence of tasks performed by the system over time. A template of this diagram type is shown in Figure 9(*d*). One unique property of action architectures is that the propagation arrows are often based on completion or performance of tasks, rather than new inputs or output flow values. This is because action architectures are generally used to represent how the system transitions between states over time.

Finally, when analyzing the interactions between specific components in a function or set of functions, a **Component Architecture Diagram** can be constructed, as shown in Figure 9(*c*). Note that component architectures may be similar to functional architectures, since they show a static view of behavioral propagation across physical properties of the system, rather than tasks accomplished over time. The main difference is that the functional view summarizes an overall behavior (e.g., locomotion) fulfilled by a component architecture (e.g., wheels on a car). Since components may contribute to multiple functions, components may be featured in multiple component architectures contained in different functions which would take on different characteristics (e.g., wheels on a car both contribute to moving the passengers as well as supporting them vertically).

These architectures represent the propagation of behavior between different aspects of the system. In general, these views of the system may be used to form an overall hierarchy of abstraction, where the functional decomposition is used to represent the overall, integrated behavior of the system, while component and action architectures are used to represent the behavior of individual functions–specifically, component architectures are used to represent the *embodiment of functions as hardware* while action architectures are used to represent the *tasks or sequence of operations defining controller and/or operator behavior*.

3.4 Modelling and Analysis Process. FRDL thus provides a unified, integrated modelling language that can be used to represent, analyze, and ultimately understand technical structure and behavior in complex systems at varying levels of abstraction to support design and analysis activities. The next subsections describe the process for using FRDL to support design and analysis.

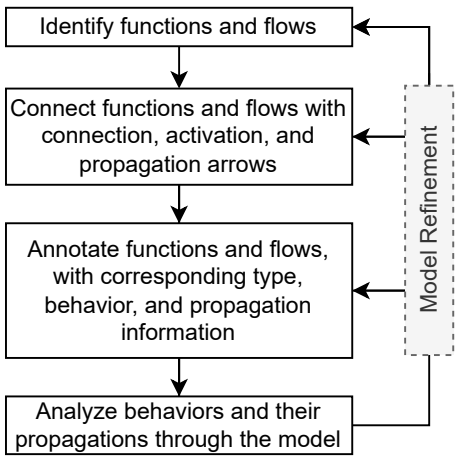


Figure 10 Process for developing FRDL-based functional models

3.4.1 Modelling Process. Developing a model in FRDL follows the process shown in Figure 10. As shown, the first step is to identify functions and flows, by identifying key behaviors to be embodied by the system (functions) and the properties shared and passed between these behaviors (flows). After functions and flows have been identified, a graph of their relationships is developed using connection, activation, and propagation arrows. These relationships are determined based on the expected behavior of each function, and how (both means and mechanics) behaviors in each function are expected to affect the behavior of other functions. The functions, flows, and relationships in this structure are then annotated with type, behavioral, architecture, and other detailed information described in the corresponding section for each element (Section 3.1.1-3.1.4 for functions and Section 3.2.1-3.2.3 for function/flow relationships). Given this structure, the expected (hazardous and nominal) behavior represented by the model can be analyzed to verify that the model embodies the desired behavior and support model refinement. Based on this process, the model

may be refined by identifying changes to functions/flows, their relationships, and annotations needed to represent the overall system behavior. Because the FRDL model is supposed to be used to directly support hazard analyses, preliminary model analysis and refinement is crucial to ensure that the model is sufficiently rigorous and valid.

3.4.2 Analysis Process. The goal of FRDL is to directly inform the hazard analysis process from the representation of system function by providing a rigorous representation of functional interactions. The hazard analysis for FRDL models thus aspires to enable the direct reliance on model constructs, structure, and annotations (as opposed to unstated analyst assumptions) to determine the effects of failures. The main process for performing hazard analysis in an FRDL is thus to:

(1) Instantiate a hazardous condition in a model entity or entities (e.g., a partial or complete loss of a function or an adverse state of a flow) at a relevant time in the scenario,
(2) Identify how this condition will directly effect the entity (including modified behaviors, damage, harm, etc),
(3) Determine how this conditions and its effects with propagate to entities connected via defined relationships,
(4) Repeat steps 2-4 for all affected entities until the system-wide effects have been exhaustively elicited.

Note that while FRDL models are meant to be explicitly used during the analysis process, they should not be treated as authoritatively correct for the purpose of analysis. In fact, performing analysis on a model can reveal underlying assumptions which were incorrect or poorly represented. Thus, it should be expected that models be refined during the analysis process to better represent behavioral propagation that is expected in hazardous scenarios. The goal of explicit modelling constructs is thus not for the diagram to *perform the analysis for the analyst*, but to help the analyst *think rigorously and systematically* about behavioral interactions, and (after the analysis) to provide a concrete basis for tracing identified hazardous effects to (instead of relying on unstated assumptions).

Another goal of FRDL is to enable one to define the structure and behavior of the system to inform hazard analysis both in early design (when one is interested in analyzing overall behavior and interactions) and carry it into the later design stages (when one is interested in analyzing the behavior of subsystems and components). Using FRDL to support the traditional hazard analysis process would thus involve going through the process shown in Figure 11 (though it should be noted that FRDL is merely a system modelling language and could, in theory, support many different types of analyses). In this process, the system may be broken down at a high level early in system development (using a function-in-context diagram) and further diagrams may be represented as the functional architecture, logical behavior (action architecture) and physical structure (component architecture) of the design are developed. Notably, unlike traditional hazard analysis processes, the representation of action architectures enables the analysis of controls, as opposed to just the physical functions. Furthermore, as successively more detailed models are developed, connected models can be refined to maintain overall model consistency and validate assumptions. This shows how FRDL can be used iterative through a design process, as opposed to only being used at a single phase (e.g., early design).

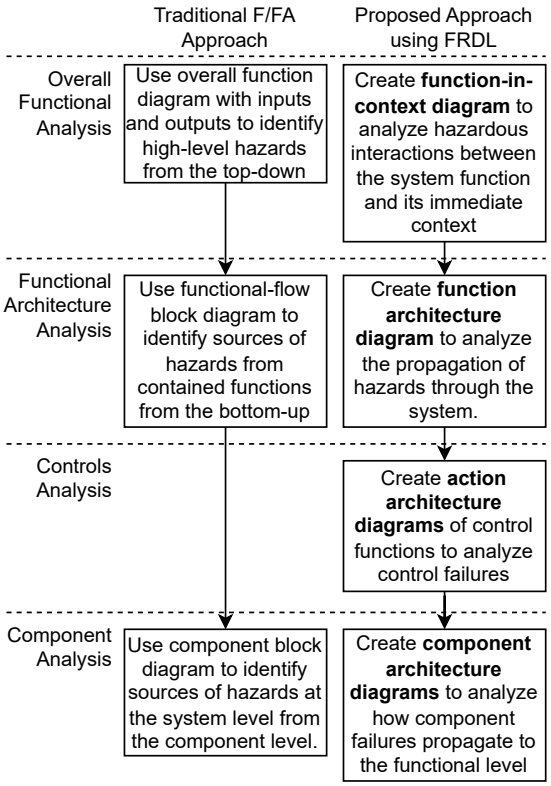


Figure 11 Proposed Hazard Analysis processes supported by FRDL.

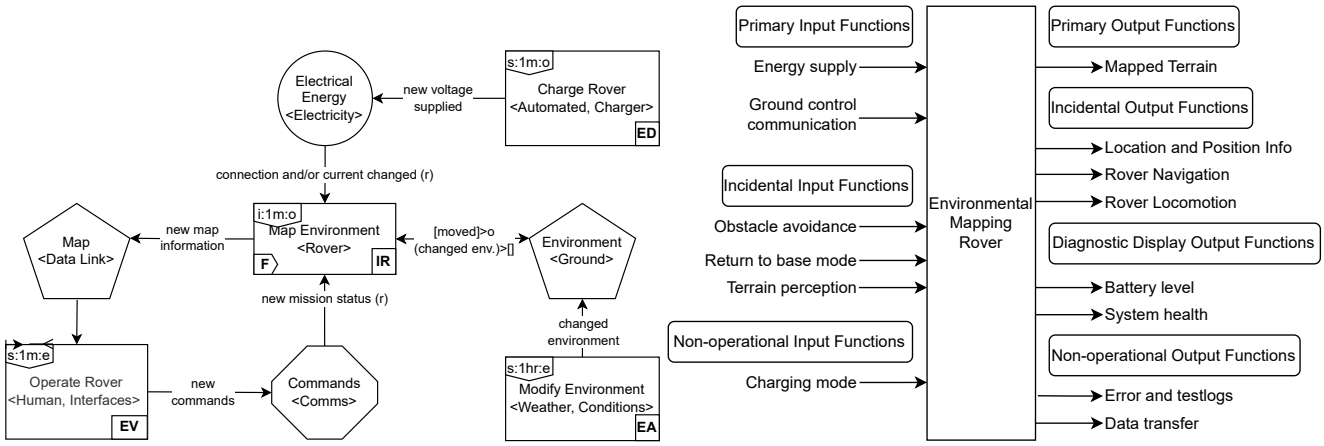
4 Demonstration

To demonstrate the use of FRDL to represent a system and analyze hazards, this section considers the application of FRDL to the hazard analysis of a semi-autonomous rover system. The goal of this demonstration was to provide a notional idea of the value and promise of the FRDL ontology and method, rather than to empirically study how participants use the method. Thus, a small example is provided here comparing the authors' analysis of a few rover faults using FRDL and the Function/Fault Analysis Procedure.

The overall mission of the rover used in this demonstration is to explore and map a remote or inaccessible region by navigating the region with a set of onboard instrumentation. The rover navigates autonomously but communicates status as well as progress back to the operator, who can in turn issue commands to override plans or operations such as modifying the trajectory or endpoint, or having the rover return to its starting position. The next subsections describe how FRDL can be used to define the function structure of a system, while comparing the use of these diagrams for hazard analysis against the Function/Fault Analysis Procedure.

4.1 Overall Functional Analysis. To analyze the functionality of the rover at a high level, a function-in-context diagram was developed to show the main interactions between functionalities provided by the rover, the operator, the environment, and external systems. As shown in Figure 12(*a*), the rover functionality (Map Terrain) interacts with the operator (Communicate with and Control Rover) via "Map" and "Commands" flows, which represent the communications channels between the rover and the operator. Prior to the mission, the rover is additionally charged via an external "Charge Rover" function, which provides an "Electrical Energy" flow to the "Map Terrain" function when connected. Finally, the "Map Terrain" function interacts with the "Environment" (represented as a flow) to create the map, while the map is being modified by various conditions referred to as the "Modify Environment" function.

A conventionally-developed function diagram for the system is provided for this system in Figure 12(*b*). As shown, while this diagram does not show the interactions between the rover system with other functions, it does provide input and output information that can be used to understand some of these interactions. One notable difference between this function and the FRDL-based diagram is



(*a*) Rover Function-in-Context Diagram (FRDL)

(*b*) Rover Function Diagram (F/FA)

Figure 12 High-level functional representations for top-down hazard analysis

Table 1 Overall Functional Hazard Analysis Excerpt

	Function	Failure Condition	Operating Phase	Causes and Mechanisms	Effects	Detection
Function Diagram (F/FA)	Environment Mapping	Incomplete Map	When on and mapping	Errors in terrain perception, obstacle avoidance, inadequate energy supply, fallacious control actions.	Mapped terrain returned is incomplete. Must send out another mission.	Operator review of mapped terrain
FRDL Function-in-Context Diagram	Map Environment	Incomplete Map	After the rover is sent commands to map and before it stops.	Inadequate charge, inability to move over environment, environment changes after mapping, termination of mission	Communicated map incomplete. Potential to degrade localization, resulting in poor locomotion.	Operator review of mapped terrain. Mission status degraded if stuck due to failed localization and inability to cover map area or reach destination(s).

that it shows more discrete inputs and outputs, whereas in the FRDL diagram, they are aggregated in flows. For example, in the FRDL diagram, "Commands" refers to "rover location and position info," "battery level," "data transfer" outputs as well as "ground control communications" and "return to base mode" in the rover function diagram. To achieve the same level of granularity as the function diagram, these flows would need to be better elaborated in detail elsewhere. Nevertheless, the FRDL diagram more visibly shows how the system interacts with its operators, environment, and technical systems in terms of means and mechanics.

To illustrate the use of the FRDL function-in-context diagram for hazard analysis, an analysis of an individual fault–the environmental mapping function creating an incomplete map–was performed using the Rover Function-in-Context Diagram and the rover Function Diagram. The results of this hazard analysis are shown in Table 1. As shown in the table, the analysis results here for each are similar, with more detailed results coming from the FRDL-based model. Notably, the FRDL-based model provides similar conditions, causes, effects, and detection results as the traditional model, but with more details in each category that emphasize the interactions and dynamics of the system (e.g., "potential to degrade localization" as an effect and "environmental changes after mapping" a potential cause), as opposed to immediate effects on input/output functionality. This is because, as shown in Figure 12, the FRDL-based model provides a representation of the rover's dynamics and interactions with external functions. This makes it possible to explicitly use these interactions to determine hazard causes, mechanisms, and effects in the context of hazard analysis

4.2 Functional Architecture Analysis. To further analyze the functions of the rover and their interactions with each other, a functional architecture diagram was developed that shows the main envisioned functions of the rover. As shown in Figure 13(*a*), the main functions are communicating with the operator, storing and supplying power (i.e., battery and power supply), navigating the environment (i.e., rover locomotion and path planning), mapping the environment (i.e., creating a map from sensor readings), and sensing the environment (i.e., sensing environmental data). These functions are connected by electrical energy, commands (i.e., control and status communications with the operator), the location and orientation of the rover, the rover's internal map, and the environment itself. It should be noted that functions could further be elicited for a rover for various expected disciplines and functions (e.g., structures, electricity distribution, thermal management). However, this scope is shown for illustrative purposes.

A corresponding function block diagram for the same rover system is shown in Figure 13(*b*). As shown here, this model provides some idea of the functions of the rover (mapping, communication, navigation, sensing, etc.) as well as their (assumed sequential) interactions. However, unlike the FRDL-based model, the mechanics of these interactions are unlisted and there is temporal ambiguity in terms of how each function is defined, and how each function leads into or causes the successive function. However, it should be noted that this diagram requires less time and effort to construct, coming at the cost of a less rigorous understanding of system dynamics and interactions.

To compare the use of both types of diagrams, a hazard analysis was performed using both considering a condition in which the ability of the rover to navigate the environment is hindered by it being stuck in a particular location. The results of these analyses are shown in Table 2. As shown, the FRDL-based model produces similar results to the traditional function block diagram, but with

more details related to the dynamics of the system. Specifically, for causes, the FRDL-based model was able to represent "unable to respond to operator commands" and "hostile terrain" as potential causes due to the representation of operator and environmental interactions in the diagram (as opposed to Figure 13(*b*), where the blocks do not have strong implied interactions). Additionally, for effects, the FRDL-based model was able to represent "unable to fully map environment" and "rover runs until battery drains" as potential effects since both "store and supply energy" and "map environment" are shown as direct connections via "electrical energy" and "environment" flows, respectively. This is unlike the traditional function block diagram, where store and supply power is represented as a "parallel" function (not necessarily interacting) and "map terrain" is represented as a preceding function (which causes effects to "navigate environment" rather than being affected by it).

4.3 Discussion. As demonstrated in Section 4, the FRDL language was able to better represent the behavioral dynamics and interactions of a rover system than a function-flow block diagram. This resulted in more detailed and physically-grounded results for hazard analysis, since the physical interactions between functions (represented by flows) were better accounted for in the function diagrams. In the demonstration of the overall functional analysis, the analysis was improved because the diagram was better able to represent the interaction between the functionality of the rover and its external environment such that the dynamic effects were able to be traced out over time (e.g., effects on locomotion). In the context of the functional architecture analysis, the analysis was improved because the "order" of functions in the Rover Function-Flow Block Diagram did not capture the dynamic interactions that navigation failures can have on environmental mapping as well as the limited power supply that were readily produced using the FRDL-based model. Thus, the FRDL-based analyses were able to better trace out hazards to their potential effects, with effects being able to be traced in more directions from their initial source–both in terms of impact to other functions as well as over time.

Interestingly, this improved representation of interactions additionally improved the analysis of hazard causes and mechanisms, which the authors did not consider until the time of the demonstration. This was found in the execution of the demonstration because tracing causal mechanisms is a similar process to tracing hazard effects, only in reverse. As a result, a variety of causal mechanisms that would not otherwise be identified were found using the FRDL-based models, from tracing the "causal mesh" provided by the model from effects to causes (instead of causes to effects). This resulted in a range of mechanisms being uncovered from elsewhere in the functional architecture, including adverse environmental conditions and control commands in the overall functional analysis and hostile terrain and inability to respond to operator commands in the functional architecture analysis. This demonstrates how FRDL-based architectures can help inform functional hazard analysis: by creating a rigorous model of behavioral interactions, the methodology providing a more comprehensive picture of both causes and effects of hazardous scenarios.

5 Evaluation

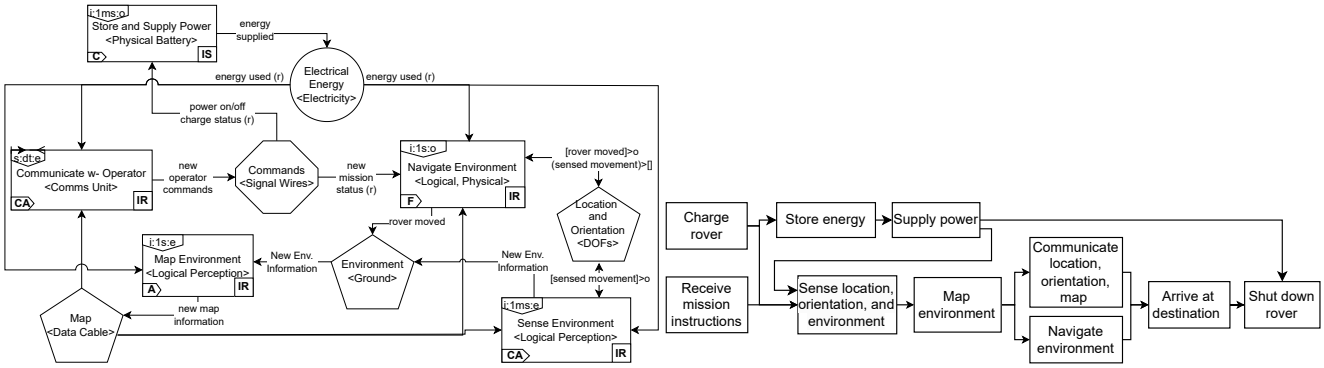
While Section 4 demonstrated the use of FRDL to support hazard analysis and provided some evidence of its usefulness compared to function-flow block diagram, it may be difficult to understand how these results situate it with a number of FHA-supporting diagrams and analysis methodologies. As such, this section aims to further show how FRDL compares with these methods as well as provide some commentary on the usefulness of the approach. While an exhaustive demonstration comparing the *use* of all methods is out of the possible scope of this paper, Table 3 provides an overview of how a variety of existing FHA-supporting diagrams represent functions and interactions, and how this subjects them to the problems presented in Section 3. Further comparison of these diagrams (with visual examples) is provided with the FRDL specification provided on the fmdtools documentation page. [2]

As shown, each of the major existing methods are subject to the problems identified to some extent, though there are some differences between methods that may make them more or less attractive depending on the goals of the analysis. The most simple of the approaches, the hierarchical functional decomposition, is the simplest diagramming method and avoids some of the issues that graph-based methods face by simply not attempting to represent behavioral interactions at all. Similarly, STAMP/STPA's relative simplicity, defined frame of reference and scope (only control interactions) similarly shields it from the complex task of supporting general hazard analysis because its stated goal is supporting accidents that happen within the control structure. Thus, while simpler diagramming approaches can be appropriate (and can often provide results quicker) for particular types of analyses, they lack the generality of analysis viewpoint that FRDL provides. However, when the task is specifically to inform the general analysis of technical function interactions–the goal of FRDL–the existing graph-based methodologies for hazard analysis (EMS-based models, Function-Flow Block Diagrams, FRAM Function Diagrams) are largely subject to the problems identified in Section 3.

It should be noted how well SysML Internal Block Diagrams (IBDs) address the problems identified in Section 3, addressing Problems 1 and 2 directly and Problem 3 partially. While SysML has been used for general hazard analysis in a number of examples in the literature, the use for FHA is somewhat more muddled. Generally, most of these methods are about the use of SysML to encourage MBSE practices, rather than to better inform the analysis of hazard propagation. As such, there has not been much demonstration about how to set up these models with an effective functional abstraction to perform analysis on, other than the creation of profiles to use [43,56]. However, most of these approaches map functions to blocks, which is consistent with the internal block diagram representation provided by Kruse et al [55]. However, it should be noted that SysML internal block diagrams are still subject to Problem 3, because their flow arrows do not specify propagation mechanics (e.g., how one block's behavior may affect another's). Additionally, because they apply a unipartite representation of connections, IBDs are subject to Problem 4, making it difficult to effectively analyzing complex interactions over a shared medium (e.g., environment, communications medium, highly coupled technical interactions etc.). Thus, while IBDs have good potential for supporting the FHA process (especially due to their support of MBSE), they lack some of the capabilities of FRDL needed to analyze particularly complex interactions between the system, operators, and environment.

Aside addressing the problems summarized in Table 3, FRDL has a broader range of potential advantages over some of the more commonly-used approaches Unlike traditional FHA approaches, the formalism FRDL provides can be used to perform controls analysis, a capability which is becoming more and more important as systems become more automated. While this analysis can already be performed using an approach like STPA [19], FRDL's unified language means that the analysis of controls can be inherently integrated with component and functional analyses, rather than requiring a separate model and process. Another major advantage of this unified view is that it means there can be direct, internal consistency between the different types of analyses that carry through the safety analysis process from the earliest stages (where a functional abstraction is needed) through detailed design. This in turn should enable a streamlined V&V process, since high-level and low-level analyses of hazards can be kept consistent throughout design, rather than having high level analyses be abandoned as the design becomes more detailed, only to then need to be brought into consistency as a part of V&V. While there are no technical capabilities supporting this approach as of yet, it should

[2] https://nasa.github.io/fmdtools/docs-source/FRDL.html



(*a*) Rover Function Architecture Diagram

(*b*) Rover Function Block Diagram

Figure 13 Rover functional models for bottom-up functional analysis

Table 2 Functional Architecture Hazard Analysis Excerpt

	Function	Failure Condition	Operating Phase	Causes and Mechanisms	Effects	Detection
Function-Flow Block Diagram (F/FA)	Navigate Environment	Stuck at location	While navigating	Drivetrain component failure, poor mapping, sensing, power supply or charge.	Unable to arrive at destination and shut down rover.	Status communicated to operators.
FRDL Functional Architecture	Navigate Environment	Stuck at location	Between operator commands to start and end mission	Drivetrain component failure, poor mapping, sensing, power supply or charge. Unable to respond to operator commands. Hostile terrain.	Unable to arrive at destination and shut down rover. Unable to fully map environment. Rover runs until battery drain (or operator shutoff), resulting in loss of communications, sensing, mapping functions.	Status communicated to operators

be noted that this unified view is possible to achieve in simulation (see: [22,40]) as well as in existing MBSE tools.

However, it is important to recognize that this approach requires an increased level of time and attention to develop the model in a rigorous way. Capturing all properties, behaviors and interactions required by an FRDL-based model can be a painstaking process that requires going back and forth with the model and the system design or concept. The aim is a more rigorous formalism that can be better used to identify and justify hazard analysis results, since the diagram represents a much larger (and more rigorously defined) set of causes and effects than before. This rigor should encourage designers to think more deeply about the system hazards early in the design process, resulting in (1) safer, more thought-out system architectures and (2) an improved ability to justify safety claims. However, it should be noted that this approach levies real costs on the design process, increasing the amount of time required to begin analysis. Thus, while these costs may be "pay off" in complex safety-critical applications, it may not be justified when the system is relatively simple and/or less critical to safety.

To close, one useful method for considering the validity of design methods is the Validation Square, which breaks the task of validating a design method into two processes–demonstrating the theoretical correctness of method constructs and demonstrating the empirical performance of the method on example problems–to support the claim that the method will be efficient and effective at achieving its proposed purpose on new problems of interest [57]. Section 3 and Table 3 demonstrate the first by making the case that analysis of FRDL-based functional architectures provide a *correct* means to trace hazardous conditions in one function into subsequent conditions and states in affected functions. Some empirical support for the applicability and performance of this method is further provided in Section 4, where it is used in the context of FHA, and is shown to help identify more effects (as well as causes) than would be found with a function-flow block diagram. For the purpose of proving the empirical performance of FRDL, this demonstration is incomplete–focusing only on a single system, a limited set of faults, and a single comparison method–and is subject to the biases of the researchers. Thus, while, it marks the first step towards demonstrating the empirical performance of FRDL, further studies are required to support this claim.

6 Conclusion

This paper highlighted some of the challenges using existing functional representations of systems to support hazard analysis. In hazard analysis standards, the representation of the system is often a block diagram or containment structure, which both provide an inadequate understanding of system interactions and behaviors for understanding how hazardous conditions propagate through the system. To address these challenges, this paper proposes the use of the Functional Reasoning Design Language (FRDL) to represent system function and interactions in support of hazard assessment. This paper then demonstrated the use of FRDL on a rover used for mapping an unknown environment, highlighting how FRDL's improved representation of behavioral interactions can improve hazard assessment by improving the ability of the analyst to trace hazardous conditions to their system-wide effects. While this demonstration is in no way comprehensive, the fact that FRDL resolves a number of theoretical issues identified in existing FHA diagrams can give one confidence in the comparative validity of the method.

This work represents the introduction of a coherent, rigorous diagramming language to support the assessment of hazards in the engineering process. However, the rover demonstration and the resulting hazard assessment was limited in scope with an emphasis on demonstrating FRDL rather than empirically studying its utility when compared to existing approaches. In future work, we hope to study the use of FRDL vis-a-vis existing FHA-supporting diagrams for identifying hazardous conditions, scenarios, and effects. Ideally, future work should empirically study if designers using

Table 3 Comparison of FRDL with Other FHA-Supporting Diagrams

Diagram	Function Representation	Interaction Representation	Applicability of Problems
(Hierarchical) Functional Decomposition per guidance in ARP-4761 [1]	Functionalities required by the system to provide its overall functionality	Part/whole relationships specified via containment arrows between functions and sub-functions	1: Flexible but lacks formal definition, 2: Broadly applicable, 3: Neither means nor mechanics specified (other than part/whole relationships), 4: N/A since graph is not attempting to represent behavioral interactions.
Function-Flow Block Diagrams per ARP-926C [7]	Task(s) or operation(s) performed by the system	Spacio-temporal order of (sequential or parallel) operations specified via arrows	1: Flexible but lacks formal definition, 2: Broadly applicable, 3: Only spacio-temporal "sequence" represented, 4: N/A due to lack of flows
FRAM Function Diagrams per Hollnagel [48]	Abstract processes or activities performed by system and its organizational context	Defined input, output, time, controls, preconditions, and resource relationships between functions specified by function ports	1: Addressed by inherent modelling flexibility, 2: Somewhat applicable, 3: Addressed by the 6 different relationship types, 4: Broadly applicable
STAMP/STPA Systems-Theoretic Accident Model per the STPA handbook [54]	Controller or controlled process(es)	Mechanics of control interaction (activation) specified by feedback and control arrows	1: Definition limits ability to decompose non-control technical functions, 2: Partially addressed by providing internal details of blocks, 3: Interaction mechanics (feedback/control) represented, but not means, 4: N/A since edges represent mechanics only
(EMS-based) Functional Models per Tumer et al. [12] and related methodologies [13]	Task(s) performed by the system to be embodied via an aggregation of component	Spacio-temporal order of operations, as well as means of behavioral propagation specified via (energy, materials, and signals) flow arrows	1: Strong basis in design theory (tasks to be embodied by aggregated components) undermined by artificial limitations (noun-verb-pairs), 2: Broadly applicable, 3: Represents combined spacio-temporal "sequence" with means (flows) but not propagation mechanics, 4: Broadly applicable due to unipartite graph-based representation
SysML Internal Block Diagrams per Kruse et al. [55]	Represented as Blocks, which are structural elements of the system that may encapsulate properties, behaviors, and classes	Means of interaction specified via connectors connecting block Ports with associated Item Flow properties	1: Multiple formal function definitions possible, 2: Largely addressed via block properties, 3: Able to represent means of interaction via Item Flow but not propagation mechanics, 4: Broadly applicable due to unipartite graph-based representation
FRDL Function Architecture	Abstract functional behavior to be embodied via an aggregation of components	Means of interaction represented via flow nodes, with mechanics of interaction represented by connection, activation, and propagation arrows	1: Addressed by delineating functions, actions, and components, 2: Addressed via block annotations, 3: Addressed by delineating Connection, Activation, and Propagation relationships, 4: Addressed with bipartite graph-based model representation

this language identify more hazards, and effects. Additionally, this language is currently merely a specification included as a part of the fmdtools repository [58], where it it used for model visualization. In future work, we hope to develop a comprehensive toolset for hazard analysis (see: [59]), including a model-based systems engineering tool for developing these models and linking them with explicit analyses (FHA, FMEA, FTA, etc). Finally, while this work introduces FRDL in the context of hazard analysis, it may additionally prove useful in a wide range of functional modelling use-cases in early design. Future work should thus study the use of FRDL to support general functional reasoning for the purposes of design.

Acknowledgments

This research was funded by the System-Wide Safety project in the NASA Aeronautics Research Mission Directorate. The findings herein represent the research of the authors and do not necessarily represent the view of the U.S. Government or NASA. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the U.S. Government.

References

[1] S-18 Aircraft and Sys Dev and Safety Assessment Committee, 2023, "ARP4761A - Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," SAE International, https://www.sae.org/standards/content/arp4761a/

[2] Tan, J. J., Otto, K. N., and Wood, K. L., 2017, "Relative impact of early versus late design decisions in systems development," Design Science, **3**, p. e12.

[3] 2012, "MIL-STD-882E," United States Department of Defense, https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=36027

[4] Joshi, A., Whalen, M., and Heimdahl, M., 2005, "Model-based safety analysis final report," National Aeronautics and Space Administration, https://shemesh.larc.nasa.gov/fm/papers/Model-BasedSafetyAnalysis.pdf

[5] Graydon, M., Neogi, N. A., and Wasson, K., 2020, "Guidance for designing safety into urban air mobility: Hazard analysis techniques," *AIAA Scitech 2020 Forum*, Orlando, FL, p. 2099, doi: https://doi.org/10.2514/6.2020-2099.

[6] Denney, E. W., 2022, "AdvoCATE User Guide," *NASA V&V Commercial Systems TC-3 Conference and Seminar Series*, Virtual, https://ntrs.nasa.gov/citations/20220009664

[7] Aircraft, S. S.-., Dev, S., and Committee, S. A., 2018, "SAE ARP926C: Fault/Failure Analysis Procedure," SAE International.

[8] Wilkinson, P. and Kelly, T., 1998, "Functional hazard analysis for highly integrated aerospace systems," *IEE Certification of Ground/Air Systems Seminar (Ref. No. 1998/255)*, IEEE Xplore, London, UK, doi: 10.1049/ic:19980312.

[9] Leveson, N., 2016, "STPA (System-Theoretic Process Analysis) Compliance with MIL-STD-882E and other Army Safety Standards," Massachusetts Institute of Technology, http://sunnyday.mit.edu/compliance-with-882.pdf

[10] 2022, "ISO-21448 Road vehicles — Safety of the intended functionality," International Organization for Standardization, https://www.iso.org/standard/77490.html

[11] Pahl, G. and Beitz, W., 2007, *Engineering design: a systematic approach*, Springer Science & Business Media.

[12] Stone, R. B., Tumer, I. Y., and Van Wie, M., 2004, "The Function-Failure Design Method," Journal of Mechanical Design, **127**(3), pp. 397–407.

[13] Jensen, D., Van Bossuyt, D. L., Bello, O., O'Halloran, B. M., and Papakonstantinou, N., 2024, "A Survey of Function Failure Identification and Propagation Analysis Methods for System Design," Journal of Computing and Information Science in Engineering, **24**(9).

[14] Kurtoglu, T. and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," Journal of Mechanical Design, **130**(5), p. 051401.

[15] McIntire, M. G., Keshavarzi, E., Tumer, I. Y., and Hoyle, C., 2016, "Functional models with inherent behavior: Towards a framework for safety analysis early in the design of complex systems," *ASME International Mechanical Engineering Congress and Exposition*, Vol. 50657, American Society of Mechanical Engineers, Phoenix, Arizona, USA, p. V011T15A035, doi: https://doi.org/10.1115/IMECE2016-67040.

[16] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2009, "Flow State Logic (FSL) for analysis of failure propagation in early design," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 49057, San Diego, California, USA, pp. 1033–1043, doi: https://doi.org/10.1115/DETC2009-87064.

[17] Patriarca, R., Di Gravio, G., Woltjer, R., Costantino, F., Praetorius, G., Ferreira, P., and Hollnagel, E., 2020, "Framing the FRAM: A literature review on the functional resonance analysis method," Safety Science, **129**, p. 104827.

[18] Zhang, Y., Dong, C., Guo, W., Dai, J., and Zhao, Z., 2022, "Systems theoretic accident model and process (STAMP): A literature review," Safety science, **152**, p. 105596.

[19] Ishimatsu, T., Leveson, N. G., Thomas, J., Katahira, M., Miyamoto, Y., and Nakao, H., 2010, "Modeling and hazard analysis using STPA," .

[20] Leveson, N., 2004, "A new accident model for engineering safer systems," Safety science, **42**(4), pp. 237–270.

[21] Hulse, D., Walsh, H., Dong, A., Hoyle, C., Tumer, I., Kulkarni, C., and Goebel, K., 2021, "fmdtools: A fault propagation toolkit for resilience assessment in early design," International Journal of Prognostics and Health Management, **12**(3).

[22] Irshad, L. and Hulse, D., 2022, "Resilience Modeling in Complex Engineered Systems With Human-Machine Interactions," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 86212, American Society of Mechanical Engineers, p. V002T02A024, doi: https://doi.org/10.1115/DETC2022-89531.

[23] 2011, "AC.23.1309-1E," Federal Aviation Administration, https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_23_1309-1E.pdf

[24] 2018, "ISO 26262 Road vehicles — Functional safety," International Organization for Standardization, https://www.iso.org/standard/68383.html

[25] 2018, "ARP1834 Fault/Failure Analysis For Digital Systems and Equipment," SAE International, https://www.sae.org/standards/content/arp1834/

[26] SC-205, C., 2011, "DO-178C Software Considerations in Airborne Systems and Equipment Certification," RTCA, Inc.

[27] Noviello, M. C., Dimino, I., Concilio, A., Amoroso, F., and Pecora, R., 2019, "Aeroelastic assessments and functional hazard analysis of a regional aircraft equipped with morphing winglets," Aerospace, **6**(10), p. 104.

[28] Meyer, L., Vogel, M., and Fricke, H., 2010, "Functional hazard analysis of virtual control towers," *11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Vol. 43, Elsevier, Valenciennes, France, Paper No. 13, pp. 146–151.

[29] Tran, V. N., Tran, L. V., and Tran, V. N., 2021, "Functional Hazard Analysis for Engineering Safe Software Requirements," *2021 4th International Conference on Information and Computer Technologies (ICICT)*, IEEE, Kahului, HI, United States, pp. 142–148, doi: 10.1109/ICICT52872.2021.00031.

[30] Nagy, B., Edwards, L., and Sivapragasam, G., 2021, "Functional hazard analysis and subsystem hazard analysis of artificial intelligence/machine learning functions within a sandbox program," *Proceedings of the 18th Annual Acquisition Research Symposium*, Naval Postgraduate School Acquisition Research Program, Monterey, CA, United States, https://dair.nps.edu/handle/123456789/4401

[31] Ericson, C. A. et al., 2005, *Functional Hazard Analysis*, John Wiley & Sons, Ltd, Chap. 15, pp. 271–289.

[32] Johannessen, P., Grante, C., Alminger, A., Eklund, U., and Torin, J., 2001, "Hazard analysis in object oriented design of dependable systems," *2001 International Conference on Dependable Systems and Networks*, IEEE, Gothenburg, Sweden, pp. 507–512, doi: 10.1109/DSN.2001.941436.

[33] Modarres, M., 1993, "Functional modeling of complex systems using a GTST-MPLD framework," *Proceedings of the International Workshop on Functional Modeling of Complex Technical Systems*, Ispra, Italy, pp. 12–14.

[34] Rasmussen, B. and Whetton, C., 1997, "Hazard identification based on plant functional modelling," Reliability Engineering & System Safety, **55**(2), pp. 77–84.

[35] Stone, R. B. and Wood, K. L., 1999, "Development of a functional basis for

design," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 19739, American Society of Mechanical Engineers, Las Vegas, NV, United States, pp. 261–275, doi: https://doi.org/10.1115/DETC99/DTM-8765.

[36] Stone, R. B., Tumer, I. Y., and Van Wie, M., 2005, "The function-failure design method," Journal of Mechanical Design, **127**(3), pp. 397–407.

[37] Sangelkar, S. and McAdams, D. A., 2012, "Creating actionfunction diagrams for user centric design," *2012 ASEE Annual Conference & Exposition*, American Society For Engineering Education, San Antonio, TX, United States, pp. 25–355, doi: 10.18260/1-2–21113.

[38] Soria Zurita, N. F., Stone, R. B., Onan Demirel, H., and Tumer, I. Y., 2020, "Identification of human–system interaction errors during early design stages using a functional basis framework," ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering, **6**(1), p. 011005.

[39] Irshad, L., 2021, "A framework to evaluate the risk of human-and component-related vulnerability interactions," https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/1257b104p

[40] Irshad, L., Ahmed, S., Demirel, H. O., and Tumer, I. Y., 2019, "Computational functional failure analysis to identify human errors during early design stages," Journal of Computing and Information Science in Engineering, **19**(3), p. 031005.

[41] Irshad, L. and Hulse, D., "Towards Early Design Modeling and Simulation of Distributed Situation Awareness," Journal of Computing and Information Science in Engineering, pp. 1–13.

[42] Hause, M. et al., 2006, "The SysML modelling language," *Fifteenth European systems engineering conference*, Vol. 9, pp. 1–12.

[43] Schäfer, M., Berres, A., and Bertram, O., 2023, "Integrated model-based design and functional hazard assessment with SysML on the example of a shock control bump system," CEAS Aeronautical Journal, **14**(1), pp. 187–200.

[44] Jiao, J., Pang, S., Chu, J., Jing, Y., and Zhao, T., 2021, "An Improved FFIP Method Based on Mathematical Logic and SysML," Applied Sciences, **11**(8), p. 3534.

[45] El Ariss, O., Xu, D., and Wong, W. E., 2011, "Integrating safety analysis with functional modeling," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, **41**(4), pp. 610–624.

[46] Zikrullah, N. A., Kim, H., van der Meulen, M. J., Skofteland, G., and Lundteigen, M. A., 2021, "A comparison of hazard analysis methods capability for safety requirements generation," Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, **235**(6), pp. 1132–1153.

[47] Ishimatsu, T., Leveson, N. G., Thomas, J. P., Fleming, C. H., Katahira, M., Miyamoto, Y., Ujiie, R., Nakao, H., and Hoshino, N., 2014, "Hazard analysis of complex spacecraft using systems-theoretic process analysis," Journal of spacecraft and rockets, **51**(2), pp. 509–522.

[48] Hollnagel, E., 2017, *FRAM: the functional resonance analysis method: modelling complex socio-technical systems*, CRC Press.

[49] Toda, Y., Matsubara, Y., and Takada, H., 2018, "FRAM/STPA: Hazard analysis method for FRAM model," *Proceedings of the 2018 FRAM Workshop. Cardiff, Wales*, pp. 1–17, https://functionalresonance.com/wp-content/uploads/2024/08/13-Toda-FRAMily2018.pdf

[50] 2018, "ARP 926C: Fault/Failure Analysis Procedure," SAE International, https://www.sae.org/standards/content/arp926c/

[51] Hulse, D., Hoyle, C., Tumer, I. Y., Goebel, K., and Kulkarni, C., 2020, "Temporal Fault Injection Considerations in Resilience Quantification," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 84003, American Society of Mechanical Engineers, Virtual, Online, p. V11AT11A040, doi: https://doi.org/10.1115/DETC2020-22154.

[52] Bondy, J. A., Murty, U. S. R., et al., 1976, *Graph theory with applications*, Vol. 290, Macmillan London.

[53] Goh, K.-I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., and Barabási, A.-L., 2007, "The human disease network," Proceedings of the National Academy of Sciences, **104**(21), pp. 8685–8690.

[54] Leveson, N. and Thomas, J., 2018, "STPA Handbook," Massachusetts Institute of Technology, http://psas.scripts.mit.edu/home/get_file.php?name=STPA_Handbook.pdf

[55] Kruse, B., Gilz, T., Shea, K., and Eigner, M., 2014, "Systematic comparison of functional models in SysML for design library evaluation," Procedia CIRP, **21**, pp. 34–39.

[56] Lai, K. K.-Y., 2022, "A Guideline for the Implementation of Model-Based Functional Hazard Assessment and its Integration with Model-Based Systems Engineering," Master's thesis, University of Toronto (Canada), http://hdl.handle.net/1807/128100

[57] Pedersen, K., Emblemsvåg, J., Bailey, R., Allen, J. K., and Mistree, F., 2000, "Validating design methods and research: the validation square," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 35142, American Society of Mechanical Engineers, Baltimore, Maryland, USA, pp. 379–390, doi: https://doi.org/10.1115/DETC2000/DTM-14579.

[58] NASA, "fmdtools," https://github.com/nasa/fmdtools

[59] Mbaye, S., Hulse, D. E., Irshad, L., Walsh, H. S., and Andrade, S. R., 2025, "Towards Computational Functional Hazard Assessment (CFHA): A Gap Analysis and Concept for Emerging Aviation Systems," *AIAA SCITECH 2025 Forum*, Orlando, FL, United States, p. 1411, doi: https://doi.org/10.2514/6.2025-1411.

List of Figures

List of Tables