A Hybrid Framework for Critical Infrastructures Interdependency Modeling, Simulation, and Analysis

David Corder Hinton
Missouri University of Science and Technology

A HYBRID FRAMEWORK FOR CRITICAL INFRASTRUCTURES INTERDEPENDENCY MODELING, SIMULATION, AND ANALYSIS

by

DAVID CORDER HINTON

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

2023

Approved by:

ABSTRACT

Flow system models, also known as flow network models, encompass vastly complex, ever-expanding problem sets which comprise the foundation for maintenance, operation, and improvement of critical infrastructures around the world. The stable operation of these vast critical infrastructures is fundamental to the continued advancement of modern society. These infrastructures are tightly interdependent and vulnerable to interruption by both natural circumstance and malicious targeting. This necessitates representation of such critical infrastructures and their multi-domain interdependencies in defense focused constructive and virtual simulation environments as a matter of national interest and security. By breadth exploration of the problem space, this work body captures the essence of many existing solutions - addressing respective deficiencies - and simultaneously draws inspiration from their strengths to define a novel framework with which modeling, simulation, and analyses in this vast problem space may be revolutionized.

ACKNOWLEDGMENTS

I extend my thanks and appreciation to Dr. Kimball, Dr. Dagli, and Dr. McMillin. It has been my great pleasure learning from you all, working with each of you through my master's studies here at S&T, and taking on this great challenge with your support and guidance. I sincerely believe that each of you has impacted me significantly such that the knowledge and lessons you've imparted are evident throughout this body of work. My thanks to my wife and family. Without your patience and support I would never have been able to complete this endeavour. Finally, my thanks to my colleagues and industry partners who have pushed me and helped me to build all of the knowledge of practice which was necessary to bring this effort to fruition.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Flow systems are at the heart of critical infrastructures and are thus the focus of this investigation and greater work body. In simplest form, a flow system model is a mono-domain representation of infrastructure which transports resources or energy. Such a model is most simply characterized by a set of nodes linked by edges. In such resource/energy flow system (REFS) models, edges unite nodes to describe the character of transfer over a finite time sample (flow). Many REFS models facilitate transfer of a sum of energy, whether in the form of raw energy (electrical, thermal), or in the form of a material phase (solid, liquid, gas). Thereby, flow systems may be united across domains by a relation of energy transfer over time, i.e., power. While this generalized description may be useful for conventionally understood systems of flow, applications of the same general principles are applied across emerging and established disciplines alike.

A well understood domain-specific flow problem is that of potable water distribution throughout a municipality. Similarly well understood is the problem of electrical power flow to an equally complex consumer set. Consumers across developed nations enjoy high quality, high reliability flow of electrical power and clean potable water to their outlets and faucets, respectively, without much consideration for how those resources reach their domestic points of interface. While these two domains are among the most appreciable, they represent a fraction of the total flow system problem space. Additional closed systems representable by conventional flows include natural gas, petroleum (crude, refined), and sewage waste. Although not universally considered in the problem space, recent investigations have additionally included systems of information/data/packet transfer (cellular networks, satellite constellations), motor vehicle traffic[1], economics[2], neurology[3], epidemiology[4], ecology[5], and chemistry[6]. Generalizing further, batch flow systems may also be considered in the problem space. Examples include logistical systems of freight, spanning multiple spatial domains (land, water, air), and may reasonably include batches of critical resources such as household goods (bread, meat, etc.) or critical energy solids (coal, lumber).

REFS model application diversity lends itself to specialization and use of conventions and techniques which vary in name and character by domain. Over time, domain-specific solutions have become ingrained in respective industries and are thus trusted as the standard by which particular REFS models are solved. Despite domain partitions, commonalities shared across domains have led to more advanced frameworks for unified modeling of energy/power systems (FUMES). Despite the growing need for such utilities, the complexity of such FUMES limits adoption particularly where fiscal considerations and limited timelines guide decision making [7]. However, FUMES are central in motivating the nomenclature and architecture of abstract software interfaces described by the novel framework defined here.

Numerous viable software solutions are available to model and simulate nearly every flow problem within its respective domain and industry, but most solutions entirely fail to consider the interdependencies among them. The labyrinthine nature of these interdependencies severely impedes traversal of the problem space. Even in the rare cases of traversal, proposed solutions are deeply complex and alien in comparison to the techniques used in industry. This thesis investigates and highlights the mounting deficiency posed by the absence of critical infrastructure interdependency modeling and the nth order effect propagation inherent to them. Then, a flow agent-system hybrid (FLASH) modeling and simulation framework extension — or more simply, a FLASH framework — is presented to enable such models, including a proposed software architecture. Inspired by industry standard solutions such as PowerWorld and PipeFlow, and unified solutions defined by established FUMES such as Bond-Graph Theory and Port-Hamiltonian Systems, the FLASH framework is proposed to leverage elements of each in hybrid composition.

By using accepted flow solutions in combination with agent-based architecture paradigms that allow energy domain interdependency containerization and abstraction, the FLASH framework achieves approachability and scalability in multi-domain flow system modeling. Under the FLASH framework, flow networks are defined and partitioned by domain such that domain terminology and character are preserved for a particular system model implementation. In this way, a particular system is solvable using conventional domain-specific techniques, making the solution procedure immediately recognizable to relevant domain experts. This increases their confidence in the results and removes barriers to adoption. Leveraging agent-based architecture, node-referential agents manage and link system element states. In doing so, agents facilitate multi-domain transformative/gyrative energy and resource dependency linkage through user-defined script behaviors. This pattern avoids the burden of multi-domain mastery inherent to — and required for — the use of FUMES.

Implementation of the FLASH framework requires an agent-based framework for simulation and modeling (AFSM). An AFSM is a general-purpose scenario and simulation infrastructure of which the complete description is beyond the scope of this work. However, descriptions of interfaces between the FLASH framework and the general AFSM are defined in Chapter 5, and existing candidates are later considered in Chapter 6. The proposed framework is described in terms of modern object-oriented software patterns. Successful implementation of the FLASH framework onto a compatible AFSM affords end-users a means by which any particular flow network may be migrated from industry domain simulation tools into the AFSM while remaining in a form consistent with domain conventions. The FLASH framework facilitates a workflow in which the user's scenario files are processed to define REFS models along with known initial steady-state values. For each agent, FLASH also extends the scenario space to support agent-defined associations between flow system elements through scripted behaviors, constraints, and relational interdependencies. These features and workflows form the AFSM basis and thus enable a general software-oriented description of the FLASH infrastructure as an extension to said AFSM.

This discussion includes review of flow system examples spanning several primary domains and remarks on the tools which support/enable flow-based analyses within each. With context established for non-trivial flow systems from several domains, a generalization is presented, and initial terminology defined. Following this, several FUMES patterns are leveraged to capture and discretize terminology and relationships for elements of the FLASH framework presented here. The purpose of this generalization and abstraction is not to redefine or unify the means, laws, and/or limiting constraints which yield solutions for each domain specifically, but rather to provide a common set of interfaces within which the discrete elements of a given domain system may be accessed from an agent's script context. The FLASH framework combines the domain-specific, system-centric conventions of industry, with the domain-unifying, component-centric conventions of FUMES to yield a pattern which enables scalable, portable, complex, multi-domain flow system interdependency modeling and simulation.

2. DOMAINS

Though the problem space spans a multitude of scientific domains, the discussion presented here selects a subset of domains in which flow analysis processes are recognized as the primary means to represent non-trivial flow systems. Particular domains are acknowledged and described to the extent required for abstraction into a common interface within the FLASH framework.

2.1. ELECTRICITY

Electrical power transmission involves the supply of electrical energy en masse to a consumer market, including households and businesses which create an electrical load — or demand, on the system [8]. This demand is met by generation of electrical energy at generation sites — collectively a fleet — which operates on an economic dispatch schedule, and which is regulated and operated directly by large energy cooperatives and/or conglomerates [9]. Energy transformations, through electrical transformers, occur at substations where electrical transmission grids meet distribution networks. This permits energy transfer over long distances with minimal I^2R losses in alternating current (AC) three-phase sinusoidal systems [10]. By this transmission and distribution infrastructure, high-reliability electrical voltage is made available to consumers at appropriate magnitudes and frequency based on locality.

Demand is then created and met when electrical devices (loads) interface with the respective consumer endpoint, drawing electrical current, and thereby causing power to flow, based on the character of the load and under the constraint of Ohm's Law. Electrical loads may be grouped into a summative perceived load for the purposes of macro scale models as described here. All implementations of electrical circuits include an observable property of impedance. In pure direct current (DC) systems, electrical impedance is degenerative — represented by electrical resistance alone — while pure AC systems exhibit pure electrical reactance [11]. This work body focuses on modeling flow system interdependencies; thus, REFS models describing electrical power generation, transmission, distribution, and dissipation are of interest. In such models, both active (resistive) and reactive components of impedance are present in implementation and accounted for in industry models [12]. Electrical junctions, or busses are frequently present within and throughout power transmission networks. For an arbitrary pair of directly connected busses, the direction of power flow is not necessarily regulated. Transmission networks are most often characterized by a number of busses and the lines which connect any two busses. Further, electrical transformers are described by a voltage-current relationship and include connections between busses and electrical generation sources such as fossil and hydro-electric power plants, solar farms, wind farms, etc. Such sources include a bus interface as well as generation characteristics and constraints such as absolute generation limits, economic dispatch equations, and ramp rates, among others [13]. While high-fidelity models of electrical systems account for the inherent variability of electrical generation in renewable sources, domain focused models of conventional power generation sites, such as a coal-fired power plant or natural gas plant, assume abundant availability of all required resources through an abstraction layer governed by models of economic operational efficiency based on purchase price of calorific fuel and cost of operation and maintenance (economic dispatch modeling).

Of particular interest in the greater conversation of flow system interdependencies is the operational nature of a coal-fired power plant which functions in an electrical system as a generation source, but functions in a coal batching transportation and logistics system as a consumer. Additionally, and of more immediate relevance, is a coal power-plant's role as a major consumer in a distilled water flow network. Pure water is required for sustainable production of super-heated steam to prevent buildup of deposits in steam piping. A purpose of this body of work is to describe a FLASH in which these various dependencies can be modeled, simulated, and analyzed.

Electrical busses are particularly important as a relative constant in an electrical network. While generation and load will vary based on a number of constraints, the connectivity of an electrical network itself remains unchanged under nominal conditions. This enables calculation of a bus impedance matrix — or the multiplicative matrix inverse there-of, a bus admittance matrix — which is used directly as an exhaustive set of impedance-based constraints on the system when used to solve system power flow by iterative techniques such as Gauss-Seidel and Newton-Raphson [14, 15]. In models with particular focus on electrical transmission, inductance and resistance are described to account for line impedance while capacitance and conductance account for shunt admittance, each being represented within the admittance matrix.

In order to solve flow about such system models, each electrical bus includes four flow variables wherein for a given bus at least two of the four variables are known. For unknown values, approximations are provided for an initial iteration, and a single iteration of power flow yields a set of variable results which is provided as input with known values and a constant bus admittance matrix into the next iteration of power flow. Within a small number of iterations for typical systems, the iterative output converges on a value set which describes a system steady-state [14]. In electrical systems models, elements required to solve power flow include Real Power (P), Reactive Power (Q), Electrical Voltage Magnitude (V), and Electrical Voltage Phase Angle (δ), with respect to a reference phase angle, usually given by a singular slack bus. AC sinusoidal electrical power poses a unique compatibility challenge in the context of a FLASH description due to the phasor representation of power and flow elements throughout the system and is thus addressed specifically in Chapter 5. Various tools are available throughout industry to solve flow in this domain, such as PowerWorld Simulator as discussed in Chapter 6.

Notably, these power flow determinative techniques are prerequisites to further system analyses such as fault analysis, fault recovery, and economic dispatch. Thus, abandoning such domain-specific conventions in favor of multi-energy-domain models inhibits further domain analyses and may be additionally prohibitive due to the requirement for system re-translation if electrical-domain analyses are required. Because the FLASH framework compartmentalizes each system to be solved by domain convention, this barrier does not arise.

2.2. FLUIDS

2.2.1. Water. Water distribution systems with various purpose and scale are similarly integrated seamlessly into nearly all modern construction, whether domestic or commercial. A societal expectation holds that safe for human consumption (drinking) water is available, and with limited exclusions, that indoor plumbing is integrated for sanitary waste disposal into a sewage waste flow system on demand. Particularly in the case of drinking water, high quality and purity standards are enforced on water treatment companies at every step from collection, to treatment, to distribution[16], without-which severe negative consequences arise [17]. This includes requirements for compliance with respect the ecological impacts of collection[18]. Distributors design and maintain drinking-water flow systems to provide high-reliability (uninterrupted) water to customers on-demand. Drinking water is distributed from treatment plants to consumers (household, commercial) via pressurized mains in conjunction with use of relatively high elevation water storage tanks, to supply system end-points [18, 19]. A single water treatment facility may be responsible for supplying the variable needs of consumers throughout a municipality, but when compared to electrical transmission networks, interconnection with otherwise independent service regions is less prolific.

While electrical transmission systems include complex (active and reactive) resistive terms, drinking water systems observe physical constraints based on pipe dimensions and material, which contribute to a set of frictional coefficients. These are typically known for any implementation and may be recalculated throughout the lifetime of the system as needed [16]. Water systems are subject to summative abstraction depending on the needs of the model, and often constitute fully contained systems-of-systems. In the case of a water flow system model which focuses on neighborhood water consumption and includes supply to a residential apartment building, accounting may be necessary for the presence of an on-site pumping station which increases water pressure to ensure water reaches endpoints on upper floors within regulation. Alternatively, a macroscopic model may simply account for the site pump as a system end-point in and of itself, with a summative consumption based on statistical or time-dependent average consumption for the building [20]. In either case, a critical consideration in a model of interdependent flow systems is that regardless of model fidelity between pump and endpoints, the pump itself must be considered as both a consumer of electrical power and a consumer (lower fidelity) or transformer (high-fidelity) of water pressure. The depth of flow inter-dependency present in drinking water regulatory infrastructure — the monitoring devices and communications which safeguard quality throughout such systems — when modeled as a flow system, may dramatically expand the interdependency mesh for near-endpoint fidelity analysis [21]

As described briefly in the previous section, high-purity, high-reliability water flow systems are also critical for operation of a particularly cross-domain element of flow systems — the coal-fired power plant. Additional water systems in such a plant include high-volume river-water primary and secondary cooling systems which regulate thermal state throughout the super-heated turbine steam system and constitute a complex system-of-systems interdependency, as discussed in [22]. Within the lens of inter-dependent flow systems, a model of a power plant built directly adjacent to a large river may reasonably assume that river-water is an infinitely available resource, but may alternatively account for the fact that the power plant also functions as a source of river water in which the volume of water leaving the plant system is both cleaner and thermally warmer than when introduced to the system. This is a case in which a single entity, a power plant, may function as a generation source across multiple domains — here electrical power, and volumetric river water, with the latter holding significance in a simulation of ecological impact on wildlife in and around the object river [23]. Further, some river water is filtered throughout internal plant systems and employed in a slag wastewater system which cools the burned residual carbon from the coal furnace and carries the slag to local site retaining ponds[24], while light-ash is routed to a filtration system and smokestack[24]. Though most atmospheric dissipation is steam, the limited particulate emission as well as the significant thermal dissipation may give rise to additional ecological considerations and amount to representation as additional domain sources for such models, in some cases even including integrated renewables [25].

2.2.2. Natural Gas. The natural gas life cycle is described by an intricate system of pipes commonly referred to as the natural gas transportation system. Contained within the greater system are a few primary pipeline categories, with each group of pipelines serving a role in the transportation of natural gas to end-point consumers [26]. The system entry point is the gathering network. Each wellhead — drilling collection site — routes pressurized natural gas into small diameter collection points in the gathering network. These gathering pipes then junction into larger pipelines before reaching a processing facility [26]. The degree of inter-dependency with other macroscopic flow systems of a natural gas processing facility varies significantly by the means used to separate the target methane from impurities. Under any purification method, an electrical domain flow inter-dependency is present as pumps and electrical equipment throughout the site produce to a significant electrical load. This electrical load is multiplied in facilities where traditional cryogenic distillation methods are used [27, 28]. A less electrically demanding method may include use of chemically engineered nanometer membranes with geometrically designed pores which are permeable to undesired hydrocarbons. Emerging technologies in nanometer filtration demonstrate the economic viability of filtration of less energy-dense impurities such as diatomic nitrogen and carbon dioxide, but the manufacture of such nanometer films is necessarily completed on site due to the sensitivity of the membrane [29]. Particular models may demand further consideration of flow interdependencies whether through continuous rate flow calculations or batch supply and demand based on prerequisite on-site manufacturing steps.

In many respects a natural gas processing facility model may require a similar fidelity of flow interdependency as that described for the electrical power plant. Under nominal conditions, a treatment facility will function as a pressure source of methane gas and may function as a pressure or flow source of other natural gas liquids such as ethane, propane, and propane accessories [30]. Focusing on its primary function, the facility supplies high pressure methane into the high connection density interstate pipeline system. This collective system is comprised of several dozen separate networks which transport natural gas over hundreds and/or thousands of miles across the continental United States and Alaska [31].

Natural Gas utility companies (utilities) are the primary consumers of the supply made available by the interstate pipeline system. Utilities — often present in economic flow models and typically owned privately or by local government — regulate and supply natural gas to end-point consumers including industrial facilities, commercial businesses, and households. Larger customers, such as natural gas power plants commonly have a direct connection into the interstate pipeline system and purchase natural gas directly from respective owners or energy cooperatives. This fact may lead to a simplification or edge case depending on model requirements but is handled under the FLASH framework regardless of the particular analytical scope or domain focus calling for such infrastructure.

2.2.3. Petroleum. Complex in form and function, naturally occurring, primarily liquid, hydrocarbons comprise a complex energy flow domain set with similarities in production to natural gas, but with derivative production output that couples heavily in both pressure and flow sourcing and batch flow with global transportation and logistics [32]. Generally, crude oil is acquired by drilling into the Earth's crust and there tapping into subterranean deposits. Multiple pumping methods are used in practice today. Exhibiting a higher calorific value than coal, and being markedly easier to transport than hydrocarbon gasses, oil has historically been positioned as a critical energy commodity and infrastructure target [33]. A modern method of production, fracking, involves pumping high pressure fluids(water/acids) deep into disjoint oil deposit fields to crack the rock that separates them, joining the deposits and making them accessible [34]. This technique is tightly coupled with a flow system of treated water, requiring a significant power of water (in both pressure and volumetric flow rate) to achieve desired results and subsequent production.

While limited flow system interdependencies as described above may be present at the site of production, the complex utility of crude is revealed during processing. In a petroleum refinery, crude oil is heated and distilled at high temperatures to separate hydrocarbons into distinct subsets with diverse utility. Liquefied petroleum gasses (LPGs) are distilled at approximately 20 °C and collected for use in household applications such as lighters. Between 30 °C and 105 °C, conventional gasoline used in automobiles is distilled. At temperatures up to 160 °C, naphtha — a primary component in plastics, pharmacology, textiles, and makeup — is produced. Further hydrocarbon separation occurs at higher temperatures up to 230 °C yielding kerosene — the primary fuel of global aviation. Higher temperature distillation up to 425 °C produces diesel for use in automobiles. Finally, high temperature distillation produces heavy fuel used in ocean-faring cargo ships as well as bitumen, a primary component of modern roofing and asphalt — a key material in construction of modern roads [35]. These various hydrocarbon groups encompass and motivate vast flow networks specific to each group. Of particular interest are flow networks for gasoline, kerosene, and diesel which are represented both as continuous flow networks and discrete batch networks with tight coupling into logistics and transportation.

2.2.4. Flow Calculation. Regardless of the purpose or fluid subdomain, this domain of flow systems is governed by a set of observable constraints such as frictional coefficients due to pipe material and/or sediments/contaminants in the fluid of a particular system, and fluid density, as well as state-based measures of potential (head) across junction points throughout the system. For a closed system in this domain — summation of volumetric flow into and out of the system is said to be equal to zero. The linear flow method may be applied to derive equations based on a zero headloss loop rule and nodal continuity with advantages in solution convergence for inaccurate initial approximations at the cost of iteration. Alternatively, the Hardy Cross method may be employed for each junction in the system to assume an initial flow volume and direction through each pipe in the system [36]. By conservation, Kirchoff's Junction Rule dictates that for any point in a system the sum of all flows into and away from the point sums to zero. This rule must hold for all initial assumptions of flow throughout the subject pipe network as well as for flows into and out of the system holistically. With an assumption made for each flow through each pipe in the system, A correction factor (δ) based on summative frictional constraints and assumed flow is then applied to each pipe in the system. This corrected set of flows is then used as input to the system for further iteration. As values of δ approach zero, the determined values of flow for each pipe converge on a solution with increasing precision for each iteration [36]. With flow calculated, relative headloss (reduction in pressure) may be calculated across junction points throughout the system using a preferred headloss correlation method such as Darcy-Weisbach[37] or Hazen-Williams [38]. Though iterative solutions in the domain of fluid flow are achievable with linear systems of equations, the computational burden grows exponentially with the size and connectivity of the network, making the non-trivial problem space approachable only through the use of computer software. Domain specific software considered in this body includes PipeFlow in Chapter 6.

2.3. TRAFFIC

Traffic flow models and typical interest problems center around methods to alleviate congestion in urban environments. In this context, terms like density, flow, and velocity, are present, and take on unique meaning as compared to previously discussed flow systems. Even so, models of traffic networks and logistics systems are governed by similar constraints. Macroscopic or network-level traffic models, are those which describe how flow and density in a road change dynamically as constraints and conditions throughout the system change (i.e., at a traffic signal in transition, and the resultant shock wave occurring in a high-density urban street) [39]. Small scale information is ignored, and descriptions of emergent congestion trends throughout urban sectors are gathered and aggregated into a macroscopic fundamental diagram [40]. With advancements in real-time monitoring, municipalities are building high-fidelity repositories of diverse traffic data, derived from traditional load-based (pressure) sensors, and increasingly optical cameras, and GPS by way of particular GPS enabled devices [41]. These datasets afford insights into relevant spatial and temporal relations to allow meaningful analysis into congestion propagation. This analysis is usable in smart transportation systems to optimize flow by identifying critical bottlenecks occurring under conventional solutions (predefined signal timing), and queuing vehicles to move in patterns which mitigate them — maximizing throughput [42].

This is achieved by partitioning urban streets into homogeneous zones and controlling inter-transfers (flows) among them, a technique known as perimeter control [43]. Several models exist in this space currently. Greenshield's model of traffic flow includes multiple equations which may be employed toward a generic, low-to-medium fidelity framework. The first for Greenshield's equations defines a simple inverse linear proportion between velocity and density. A negative parabolic then describes the relationship between flow (vehicles/unit time) and roadway density. Whether considering traffic flow as a regional aggregation or as individual flows among specific streets, a set of parameters define a given flow problem, and a shared set of rules may be used to solve them. A traffic flow problem assumes that for an observed network, a number of vehicles entering the network is equal to the number of vehicles leaving the network. Further, for a given intersection within the network, the vehicles entering equal the vehicles leaving [44]. Based on these rules alone, some or all unknown flows throughout a traffic network may calculated via a system of linear equations derived on said rule basis. In many cases this is sufficient for this domain in particular because flow rate information is abundant or may otherwise be gathered by observation. This is compatible with the FLASH framework to be proposed. However, this method builds only a partial representation of the system. Particularly, it lacks the detail necessary to predict conditions at other times; thus, a more complete description follows for more complex/temporally sensitive models.

Building on the above method and rule set, a coefficient of resistance may be determined for a given road, in a given direction, by analysis of known data. Said coefficient relates to physical attributes of a roadway under known constraints on flow and potential. In many cases, potentials are not initially defined under the flow method described above. In this case, a lowest relative reference potential can be assigned to a particular intersection. This reference is one in which internal flows enter and which external flows leave the intersection. Vehicles can then be said to travel high potential intersections to low potential intersections as a function of road resistance. Resistive factors include number of lanes, active construction work, as well as age and maintenance conditions of the road such as debris, potholes, etc [45]. With calculated representations of temporally static resistive coefficients and temporally dynamic intersection potentials, an additional constraint may be applied to closed paths throughout the system stating that the potential around a closed loop does not change. This additional constraint parameter set allows otherwise free (unknowable unknown) flow values remaining from the flow-only approach to be approximated throughout the model along with unknown intersection potentials — as long as the number of unknown values of flow and potential are summatively less than or equal to the number of linearly independent equations created by the constraints — by a linear iterative approach in which unknown intersection potentials and road flow rates are approximated within constraint boundaries. Iterations then converge on a constraint bounded solution, where the output of each iteration is a delta to be added to the previous iteration input, and deltas across iterations converge on zero [44].

This approach provides complete intuition not only into where and when bottlenecks occur throughout the network, but also why they occur, and is useful at a macroscopic level to aid decision makers in respective resource prioritization between creation and maintenance of new and existing infrastructures. This intuition is immediately applicable to microscopic systems flow such as logistics networks which are embedded into greater traffic systems-of-systems spanning commercial spatial domains (air, land, surface/sea) and in which individual units throughout each system depend directly on multiple processed petroleum systems and increasingly, particularly in the case of land vehicles, electrical transmission and distribution systems [46]. A particular case of interest is the direct circular dependency between the final batch flow component of gasoline distribution, wherein gasoline tanker trucks transport batches of gasoline from processing facilities to gas stations by way of traffic systems, while simultaneously depending on gasoline to function. A similar inter-dependency holds for kerosene transportation via aircraft. First order failures of such a system propagate in short order into mass transportation and logistics systems at second order. Depending on the extent of interruption, second order impacts may bear globally measurable consequences at the nth order and have particular relevance to aerospace-based defense vehicles and infrastructures[47], this criticality is transparently evident so much so that any campaign level simulation may reasonably be considered incomplete if such critical resources are assumed to be abundant.

2.4. THERMAL

Thermal flow systems may be unified across domains by relation of power and energy. As is the case with the domains described above, thermal energy flow modeling is constrained into a set of known physical principals which bring about a set of finite systems of equations in a given model. The domain of thermal energy is recognizably more complex in many regards; however, any non-trivial model of an implementation in the problem space implies rules analogous to many of the flow system rules described above, in addition to rules derived by the Second and Third Laws of Thermodynamics [48]. These may not be required for lower fidelity thermal models which are not the primary domain focus in a multi-domain model. In such case, point heat sources with perfect free-space insulation and well-defined thermal conductors may permit degeneration such that a solution by the First Law of Thermodynamics in isolation is achievable [48]. It is recognized here that the complexity in a non-ideal model requires use of advanced methods in thermal dynamics modeling. It is not; however, the purpose of this work body to describe such processes. Rather, the FLASH infrastructure simply guarantees an abstract interface wherein such heat transfer analyses may be captured in a hybrid system-nodal form. In this way, system composition enables computation of flow about the system while nodal composition enables simultaneous cross-domain interdependency linkage.

3. DOMAIN GENERALIZATION

While each domain is bounded by vastly different physical constraints, an emergent pattern spans throughout and allows each system, across every flow domain, to be generalized. With generalization comes loss of particular focus, thus while for some domains, relevant generalizations of a particular variable set may come in high-fidelity and/or high availability, other domains may require that assumptions are made to derive generalized values from component factors. Similarly, while generalized results and calculations of a particular value set may be immediately valuable for a given domain, other domains may have little to no use for them directly or may require definition of domain-specific constraints for more complete analyses. A requirement for cross-domain uniformity/compatibility demands that the generalizations defined in this section are expanded upon in the FLASH framework, primarily where cross-domain energy transformation/gyration is relevant.

3.1. POTENTIAL

Potential, a referential property which compares two points in a network, is the first domain independent property generalization observed for all flow systems described in the previous section. While in some cases potential is provided as an explicit differential, more often a single value is specified, with an implicit comparison assumed between reference points.

In electrical systems. Potential is measured in units of electrical voltage V, and in most cases, when provided as a single value, is compared against a known reference — typically electrical ground, represented as a zero voltage for all models. A difference in potential is referred to as a voltage drop (ΔV) and is typically measured across a singular electrical component or across an aggregated multi-component representation.

Potential in fluid systems is captured by pressure. Units of pressure vary by application and locale (PSI, BAR, etc.) as well as unit reference itself. Invariably, the formal measure of potential in a fluid system is pressure. However, pressure itself is often referred to as head, while pressure differential across junctions in a pipe network is commonly referred to as headloss [37]. In many closed systems, a referential pipe junction or endpoint is specified for comparison throughout the system, while in open systems, atmospheric pressure serves as reference at points of absolute outlet. Fluid systems are additionally governed by transitions in material state which constitute an inter-dependency in flow systems as a result of changes in physical characteristics, such as the property of a gas to expand and compress with respective increases and decreases in temperature and respective decreases and increases in pressure; where a combination of these conditions may result in a material phase shift as a given gas liquefies when below its two-dimensional boiling point. An example of both systems-of-systems and interdependent systems is the state transition of purified water in power plants (nuclear, coal) into super-heated steam. This steam, which spins a differential turbine, is connected roto-mechanically to an electrical generator. This constitutes multiple cross domain energy transformations/gyrations and depending on the focus of a particular model may warrant explicit definition [14]. Where this is the case, the proposed FLASH framework facilitates such a definition for use in simulation.

Measurement of potential in a traffic network is less critical for analysis of traffic congestion in many typical models due to the abundance of measured flow information, but is no less present in principle. By its relative nature, potential in a traffic network is associated with individual road intersections as well as regional aggregations of homogeneous urban zones, in each case always in comparison to an applicable reference potential.

3.2. RESISTANCE

A generalized description of the property of flow systems to inhibit flow itself is resistance. Characterized by the same name in DC electrical circuits, resistance is measured in unit ohms (Ω) and is a quantitatively measurable property inherent to all electrically conductive materials. While most consumer electronics employ resistors as discrete components, the electrical domain becomes uniquely complex at the scale of electrical power transmission and distribution — and generally in the case of all AC electrical circuits. While DC electrical system resistance is described completely by simple ohmic resistance, AC system resistance is captured via a combined representation of both electrical resistance — the tendency of an electrical component to inhibit direct charge flow — and electrical reactance, unit ohms (Ω) — the tendency of an electrical component to inhibit the change in electrical charge flow. These components are then represented as a complex number wherein resistance is the real component and reactance is imaginary. This combined representation is termed electrical impedance, unit ohms (Ω). This body of work focuses on interdependencies between and among flow systems. Therefore, AC power transmission systems are of primary relevance; and so a naming conflict is present which must be disambiguated. Thus, for an abstract/generalized representation of an AC electrical system, the general term/property of resistance captures the complex term/property of electrical impedance.

Fluid systems are simpler with respect to resistance. For an arbitrary network of pipes, resistance describes the domain property of friction. As a kinetic system property, friction inhibits the movement of a material across a surface (and similarly, through a medium). For this discussion, applications of interest include fluid resource transfer through complex pipe networks. As pressure and friction are critical to the fundamental analysis of pipe networks, coefficients of friction are provided by pipe manufacturers for a given material and pipe diameter [49]. This infrastructure ages over time, resulting in changes (usually increases) in the resistivity of a particular pipe material. However, changes are slow and may be recalculated for a given implementation. In many cases manufacturers additionally provide time dependent frictional approximations derived from existing data [49].

Measure of resistance in a traffic network is more difficult to quantify, and established units are not observed widely throughout the domain. However, as is the case with potential, resistance is present in traffic networks and particularly for individual roads is positively correlated to the age and surface conditions of the road (bumps, potholes, loose rock/asphalt, debris). A primary negative resistive correlation then is the number of regular motor vehicle lanes which make up the road. A regular posted speed-limit may affect overall resistance dynamically when compared to the likely velocity that drivers would otherwise observe based on other road and environmental conditions. The results of maintenance correlate negatively with resistance once complete, but while in progress, dramatically increase resistance as drivers slow their vehicles per reduced speed limit signage and enforcement to lower the risk/consequence of hitting construction workers. Construction doubly inhibits traffic where lane closures reduce the parallel capacity on a given road and where reduced speed limits are enforced [45].

3.3. FLOW

Flow describes the movement and/or transfer of an object, entity, or quantity from one spatial point of reference to another across a defined medium, over a unit time. Examples are extensive and in many cases are not conducive to complex systems models nor interdependencies. Interesting and relevant cases are similarly extensive; however, but not described completely in this body of work. Some cases of potential interest and application for use in a FLASH implementation include those described here as well as models of flow of communications and information over a digital or analog network[50], propagation of a virus through a population and/or across aggregated populations[4], and ecological models of carbon transfer networks among forest ecosystems[5].

Flow in electrical systems may describe one of multiple related properties, those being electrical current and electrical power, which each vary depending on the circuit context, being either DC or AC. In direct current systems, the singularly directed movement of electrons (electrical charge) across a conductive medium, usually a metal such as copper, measured over a unit time is captured and represented as direct electrical current or DC, and is measured in unit Amperes, or simply Amps (A). This describes charge flow in a DC system. In such a system, delivery of electrical current through/across a compatible electrical load is useful from the perspective of the device interface, which will specify a constant operational voltage, rated current limit and often the approximate current drawn for static loads. Limitations on current through a given interface (circuit) serve to protect connected devices and delivery infrastructure alike [51]. Observably, when current limits in a given circuit are exceeded, the result is that a protective circuit element such as a circuit breaker or fuse will trip or melt respectively, opening the circuit and immediately halting supply to the set of devices which have collectively exceeded the rated current. This is the function and behavior of an electrical breaker box, often present in residences and commercial businesses as a point of interface between higher voltage/current capacity electrical distribution infrastructure external to the building. From the perspective of electrical devices however, most applications of electrical energy transfer rely on the determination/calculation/knowledge of real electrical power — a linear product of voltage and direct electrical current — as a primary operational specification for electrical devices/loads. Knowledge of the electrical power absorbed by a load is directly relevant to the operational character of the load, and additionally offers insights into the economic operational cost there-from.

Flow of electrical power and current in the context of an alternating current, AC, system is similar in many respects to DC flows, but includes additional complexities which affect each of the properties described in this section. Alternating electrical current, AC, is an operational specification on electrical systems. In an AC system, electrical charge moves through a medium — again usually a metal such as copper — over time in a periodic sinusoidal pattern. If a singular free electron in such a system were observed during regular operation, an observer would note that the physical spatial movement of said electron follows both the positive and negative curvature of a sinusoid when measured over time. The electron itself moves a forward distance through the medium before eventually slowing, and ultimately reversing direction, repeating this pattern continuously throughout regular system operation at a usually constant frequency, with modern systems operating at either 50 or 60 Hertz (Hz). This measurably results in a zero average charge flow across an electrical endpoint. However, this is where flow of electrical power becomes relevant. Although a net zero charge is delivered across the load, electrical power is in fact delivered and thus motivates a significant focus on power dissipation (power flow) as the primary flow of interest about electrical applications in critical infrastructure. Noted previously is the added complexity inherent to the operation of AC electrical systems. An exhaustive description of the underlying physics which govern these complexities is well-established and significantly exceeds the scope of this discussion. Notably, AC systems are subject to complex representations of power and resistance as well as phasor representations of voltage and electrical current. As noted previously, these complexities derive from the tendency of a conductive medium not only to resist direct charge flow, but also to inhibit the change in charge flow as a result of the opposing magnetic field created by the movement of electrical charge [14].

Acknowledgment of these physical laws complicates electrical domain flow system properties, relationships, and analyses. Similar impacts on voltage and resistance representations (phasor voltage, electrical impedance) are discussed previously in this chapter. As with voltage, alternating electrical current is conducive to a phasor representation wherein the angular component of current is referential to that of voltage and for a given system is said to lead (capacitive) or lag (inductive) the voltage phasor. The magnitude of both phasor voltage and current may be calculated to reflect a root mean (average) squared, RMS, or effective value over the period of the sinusoid as a DC circuit equivalence constrained by electrical power dissipation across a load. Intuitively, with complex electrical impedance, and phasor representations of both voltage and current, an AC system then gives rise to a complex and/or phasor representation of electrical power. As with a DC circuit, instantaneous complex power is the product of phasor voltage and complex conjugate of current — equal in magnitude, but with a negative angular component. This results in a complex power which is composed of real and reactive components which are represented as a phasor and/or complex pair depending on the required analysis. In phasor form, the magnitude of complex power is known as the apparent power and is represented in units of Volt-Amperes, while the phasor component is termed the power factor angle and provides insight into the operational efficiency of the system. The orthogonal components then are described by load characteristics. The real component of power measures power dissipation across a load and depends directly on the resistance of the load. The imaginary component, termed reactive or quadrature power, measured in unit VAR, measures reactive power dissipation across the load and depends on the reactance thereof. This complex representation of electrical power, which is fundamentally conducive to electrical flow system analyses, is of particular importance in consideration of practical applications of interdependent flow systems in modern critical infrastructure. This power flow is the governing character of electrical transmission systems that electrify virtually all consumer and industrial loads, referred to holistically as, 'the grid.'

While charge flow (current) is of component importance to power flow in electrical transmission and distribution systems, the fluid domain analog — direct resource and material transfer — is the fundamental focus in fluid system flow analyses. While further analogs may be drawn from AC electrical systems in principle into fluid flow systems — parallels exist between electrical reactance and fluid inertia/momentum — applications of such implemented systems are not prolific in fluid flow systems and thus while acknowledged and compatible with the FLASH architecture are not described here in detail. In fluid systems, flow involves the physical spatial movement of matter (usually gas or liquid) from areas of high potential (pressure) to areas of low potential. Volumetric fluid flow is measured as the quantity of fluid, assuming uniform or calculated average density where applicable, passing through a cross sectional area of a container — typically a pipe or channel — over a unit time. The quantity of fluid leaving the system from an endpoint over the duration is then measured for a variety of reasons, one of the most prolific being the translation of a delivered fluid quantity as both a purchase of a resource and service. This is a typical interaction for residences and commercial buildings consuming water and natural gas, each motivated by a variety of reasons, with fluid quantity being assigned a currency-specific economic value. In this way, a volume, such as liquid gallons or cubic meters of uniformly pressurized gas, can be purchased by the end user and billed by the provider over a period of time, often monthly. A similarly common and relevant interaction is the unit transfer of gasoline from a gas station pump into a motor vehicle. The measured volume of liquid leaving the station flow system is determined by measuring the flow rate over a duration. This volumetric quantity is purchased, typically in unit gallons at a market driven price per gallon. In similar form to the requirements for availability of electrical power, these resources are critical to the regular function of modern society. Customers expect a safe, clean stream of drinking water to be available at the tap whenever desired and expect a consistent flow of natural gas to be available on demand for residential heating and a variety of culinary applications. Fluid flow observes a simple numerical representation like other properties throughout the domain, leading to robust methods in arbitrarily complex network analysis based on systems of independent linear equations.

Traffic flow bears similar importance to that of fluid flow. In a traffic network, flow is an average measure of vehicles passing a given point on a roadway over a certain unit time, typically hours or minutes, depending on the application. This rate is then comparable with other flow rates throughout the system to offer insights into existing bottlenecks and allows models to be built based on existing conditions to then be modified and simulated to understand likely results of changes. Efficient and high-quality proposals for new infrastructure and maintenance may then be provisioned to stakeholders to fund construction projects to reduce bottlenecks and otherwise benefit, improve, and optimize flow throughout the object traffic network. While vehicle flow rates are in reality derived from whole numbers, average measures over intervals result in floating point representations of traffic flow, including negative rates where a single bi-directional road is observed over an interval with a reference direction opposite that of the overall summative flow between the two directions. Traffic flow as a flow system property is, in many models, central to reaching a solution, in large part because and data is abundantly available by observation. As noted previously, this leads to many analyses which require partial flow data alone to determine flow throughout the system, reducing or entirely removing the need for representations of resistance and potential throughout the system as a requisite to derive a meaningful solution. In such case, suffice to assume trivial or uniform values of resistance throughout the system such that underlying methods of calculation, while potentially degenerative, remain functional.

4. EXISTING MODEL FRAMEWORKS

A number of models have been proposed with the goal of bridging cross-domain systems interactions. The cases considered do so by considering the fundamentally shared property of energy, unit joule (J), and its temporal derivative — power, unit watt (W) — empirically measurable across all energy domains. These model frameworks provide means for establishing the governing equations about unified dynamical systems. Elements and terminology defined by these frameworks are captured here as underlying motivation, principal, and for consistency in terminology used in the FLASH framework to be described.

4.1. BOND GRAPH THEORY

Originally proposed in 1961 by Paynter[52], bond graph theory has been continuously developed internationally and proposes that through means of dynamical system analysis, determination and resolution of differential equations to describe and predict the character of complex, multi-domain energy systems is possible [52]. However, typically the non-trivial task of determining the differential equations is a human responsibility, where the calculation of a numerical solution when provided those differential equations is delegated to a computer system for evaluation. Bond graph theory is a proposed method by which the job of deriving the differential equations for dynamical systems themselves is made possible by a pre-defined algorithmic procedure.

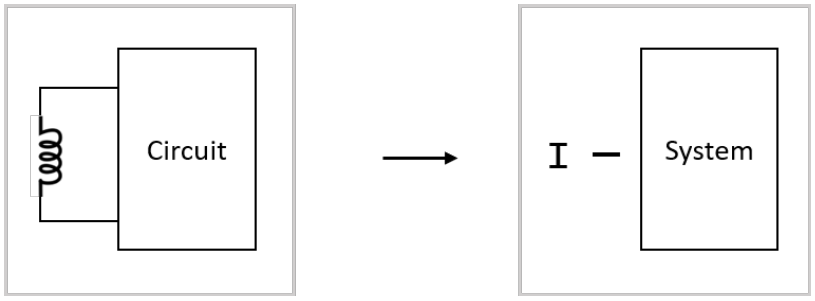


Figure 4.1. Translation from (Electrical) Domain Notation to Bond Graph Notation

In order to do so, the essential characteristics and constraints of the system must be conveyed as input unambiguously to the processing engine, thus a language must be defined to allow system characteristics to be translated for computer processing, equation derivation, and evaluation [53]. An initial example of such translation is provided in Figure 4.1.

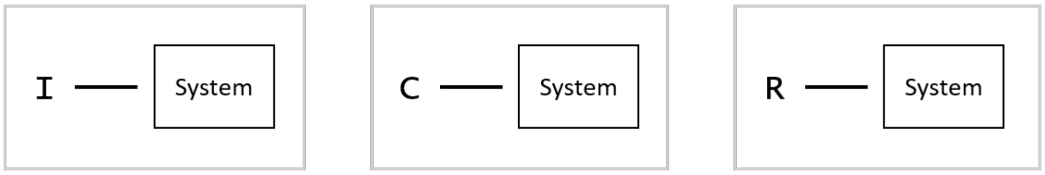


Figure 4.2. Simple Bond for Inertial, Compliant, and Resistive Element

The essential idea is that any system is composed of a set of fundamental basic elements. Though these vary in implementation by energy domain, they may be generalized and unified across energy domains by their behavioral character therein. These elements are then composed in series and parallel combination to produce a given system representation. Interactions across system elements are then united on the understanding that the fundamental exchange/transfer of power governs all. Power is composed then of the product of two variables — an effort variable and a flow variable, as described in chapter 3. The effort variable in an electrical system is the electromotive force, EMF, or voltage, while the flow variable is electrical current — the movement of electrical charge. Similarly, in a translational mechanical system, the effort variable is the force, while the flow variable is velocity. For rotational mechanical systems, effort corresponds to torque, while rotational velocity maps to flow. Relating further to magnetism, the magneto-motive force, MMF, or magnetic potential is the effort variable, while the variable corresponding to flow is the rate of magnetic flux. Bond graph theory proposes that for every dynamic domain there is always an effort variable and a flow variable and categorizes each in a group of information variables. Paynter suggests then that if the character of these three variables can be communicated to a computer for within and throughout the elements of a given dynamical system, this is sufficient for the computer to derive the governing differential equations about the system. Further, he suggests that this information can be communicated pictorially by means of a bond graph which represents only the exchange of information and power between system elements [52].

A single bond is sufficient to represent the relation between an isolated system element and the rest of the system, of which the singular element does not have knowledge — depicted in Figure 4.1. In principle, this notion motivates the nodal/elemental component of the FLASH architecture proposed in Chapter 5.

Discernible by review of the domain relation of inertial elements — such as the inductor in electrical circuits — flow is shown to be the integral of effort multiplied by a function which may in some cases (linear inertial elements) degenerate to a constant. This is captured symbolically in Equation (4.1).

$$\text{effort} = f\,\frac{\mathrm{d\,flow}}{\mathrm{d}t} \qquad\qquad \text{flow} = f\int_{-\infty}^{t} \text{effort}\,\mathrm{d}t \tag{4.1}$$

This relationship, shown in Figure 4.2, describes the bond or character of the inertial element in relation to the rest of the system. A similar relationship is evident in a number of energy systems and described in general terms in bond graph theory for compliant elements of a system (shown in Figure 4.2). This relationship is characterized in opposing form to that of the inductive element, where effort is described as the integral of flow and the product of some function which may in some cases (linear compliant elements) degenerate to a constant. Equation (4.2) captures this element character.

$$\text{flow} = f\,\frac{\mathrm{d\,effort}}{\mathrm{d}t} \qquad\qquad \text{effort} = f\int_{-\infty}^{t} \text{flow}\,\mathrm{d}t \tag{4.2}$$

This relation describes the character or bond of the inertial element in relation to the system. Finally, a depiction of the resistive element bond is shown in Figure 4.2 and captured symbolically in Equation (4.3). The character of the resistive element is such that the effort is equal to a function, which may degenerate to a constant, multiplied by the flow. This can be inversely described such that the flow is equal to the product of an inverse resistive function/constant and the effort [52].

$$\text{effort} = f \cdot \text{flow} \qquad\qquad \text{flow} = \frac{1}{f} \cdot \text{effort} \tag{4.3}$$

Bond graphs also include a description of source-type elements. Notation is provided in Figure 4.3. The first source-type element is the effort source, a system element which independently controls the character of the effort in a connected bond, but in this way, the effort source does not determine the value of the flow variable.

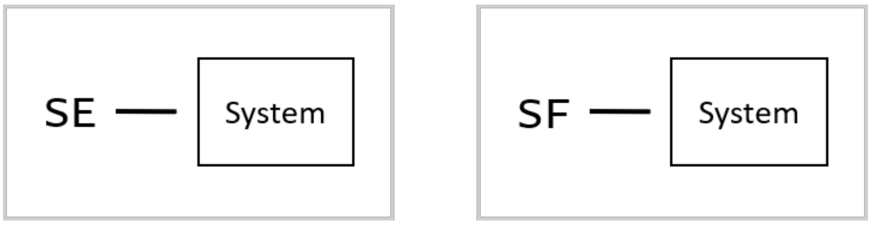


Figure 4.3. Bond Graph Source Elements

Rather the character of the flow variable is decided by the rest of the system as a whole. Based on the fact that the flow information is not decided by the effort source element and by the power relation to effort and flow that power also cannot be determined by the source of effort alone. An additional source-type element is the flow source (Figure 4.3). An element of this type independently controls/determines the character of the flow information in a connected bond. However, in this way, the flow source does not determine the value of the effort variable. Rather the effort and, in turn, the overall power is determined in combination with the rest of the system [52, 53]. Importantly, for each source element type, determination of the respective characteristic allows the source the flexibility to set a value dynamically, such as by a function of other arbitrary measures of input character. This flexibility lends itself to energy cross-domain interdependencies.

This set of elements allows for the assembled representation of simple systems with only the addition of unifying junctions. Bond graphs include two forms of junction. The zero (or p, in alternative representations) junction unifies/equalizes the effort information across all bonded elements. In other words, a zero junction receives effort information from one and only one connected bond and applies that effort to all other connected bonds. Illustrated in Figure 4.4, this principle relates to parallel flow system elements and is summative on flow about the junction.

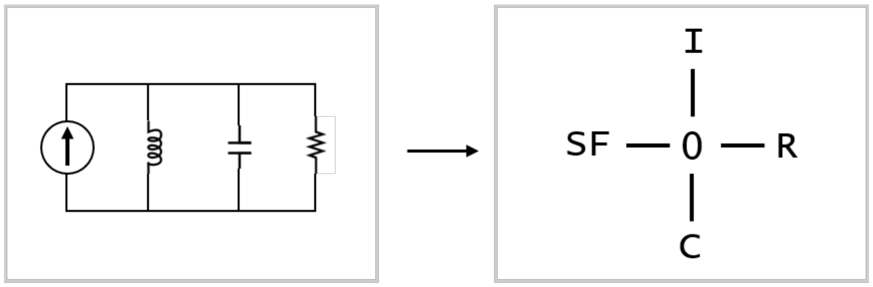


Figure 4.4. Bond Graph of Parallel RLC Circuit

The one (or s) junction unifies/equalizes the flow information across all bonded elements. In other words, a one junction receives flow information from one and only one connected bond and applies that flow to all other connected bonds. Presented in Figure 4.5, this principle relates to series flow system elements and is summative on effort about the junction.

The causal stroke, discussed below, must then appear, in relation to a given zero junction, on the junction side of only one bond, that of the effort causing element. Said bond is termed the strong bond for such a junction. Similarly, in relation to a given one junction, the causal stroke must appear on the junction side of all but one bond, that of the flow causing element. For such junction, said bond is termed the strong bond [52].

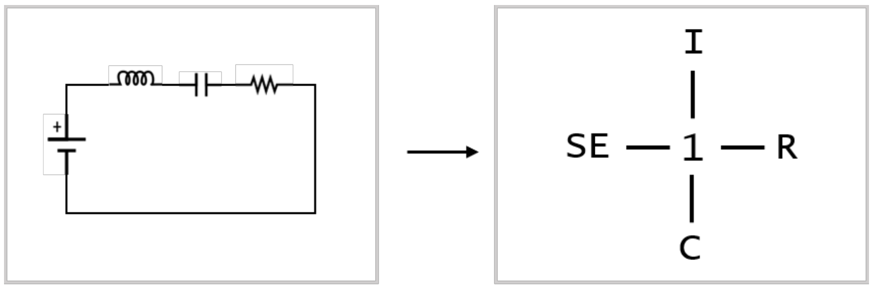


Figure 4.5. Bond Graph of Series RLC Circuit

In order for the construction of differential equations which govern such a system an additional concept of the origination and propagation of information throughout the system is needed. In effect, directionality must be defined for bonds throughout the graph. For disambiguation, this additionally requires numeric assignment to each of the bonds. Because each bond does not singularly represent either effort or flow, but rather power, the directionality of bonds in a bond graph is decided by the known or expected positivity.

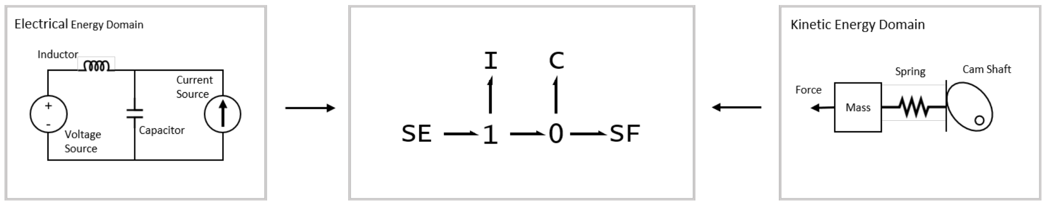


Figure 4.6. Identical Directed Bond Graph for an Electrical Circuit and a Kinetic Machine

Illustrated in Figure 4.6, positivity is generally known to be away from source elements, denoting that sources generate power. Similarly, that resistive elements are understood generally to be those which absorb or dissipate power from the system for a given domain. In the case of inertial and compliant elements; however, the direction of power flow is less clear, as for each element type, the element instance is at some point absorbing power from the system and at another point returning power to the system. The international convention for power directionality then is understood to be that which is absorbed by the element. Power directionality between bonds is arbitrary for computation and evaluation of system dynamics, but must be referred to post-evaluation to derive domain meaning for a particular time and element [52, 53].

The element set described and bond character about said elements is sufficient to describe many simple systems. Specifically, systems which include only single bond elements, or elements in which a single bond is sufficient to describe and characterize both flow and effort information about said elements. More complex systems; however, include elements which are not fully characterized by a single bond.

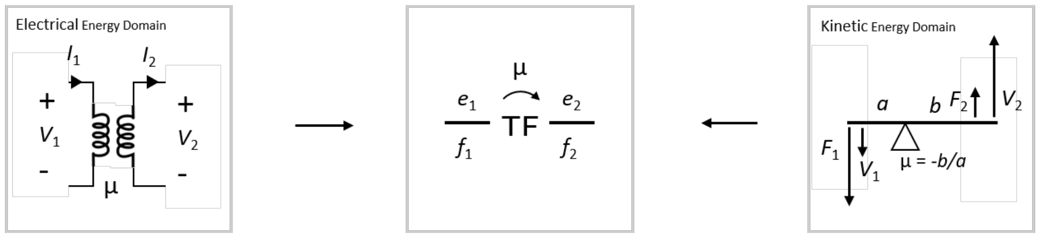


Figure 4.7. Bond Graph of Electrical Transformer and Kinetic Lever

One such element (shown in Figure 4.7) is the electrical transformer, that is, an electrical component in which inductive coils are electrically isolated, but magnetically linked such that voltage across the driving coil is transformed across the magnetic boundary to induce a secondary, and in some cased tertiary, voltage at an implementation specific transformation ratio corresponding to the number of physical coil turns in the electrical conductors on either side of the transformer. Assuming a nominal closed-circuit context, a flow applied to the driving electrical transformer (either AC or DC) induces a transformed flow in the secondary coil. This transformation is such that, for an ideal/lossless transformer, power injection into the driving/primary coil is equal to power in the secondary coil.

$$e_2 = (1/\mu)e_1, f_1 = \mu f_2 \tag{4.4}$$

Described symbolically in Equation (4.4), the secondary flow (or effort) is the product of the primary flow (or effort) and the, usually constant, transformation ratio μ. Similarly, the secondary effort (or flow) is equal to the product of the multiplicative inverse of the transformation ratio and the primary effort (or flow). In this way, the product of the primary information variables is shown to be equal to the product of the secondary information variables. Although both information relational forms are described, convention is such that the transformer relation in primary form (not the multiplicative inverse) links the flow variables. In the domain of rotational kinetics, a gear (or gearbox) is a transformer of torque and rotational velocity, while in translational kinetics, a lever (shown in Figure 4.7) or pulley system functions as a transformer of force and linear velocity [53]. For any transformer element, the bond graph representation includes two bonds about the element, each carrying primary and secondary effort and flow information respectively, and an indicating arrow carrying the transformation ratio about the element. This domain internal relation is directly captured in Chapter 5 within description of the general transformer element interface for use within the FLASH framework.

It is shown that the function of a transformer is to convert flow into flow, and effort into effort each in relation to a ratio μ. Bond graph theory additionally includes a description of the gyrator, as depicted in Figure 4.8 — an element of energy transformation which converts flow into effort and effort into flow.

$$e_2 = gf_1, f_2 = (1/g)e_1 \tag{4.5}$$

Following convention, Equation (4.5) symbolically describes gyrator operation. A gyrator is characterized such that the secondary effort is the product of the gyration ratio (g) and the primary flow. Similarly, the secondary flow is the product of the multiplicative inverse of the gyration ratio and the primary effort. Observable then, the product of the primary effort and flow is equal to that of the secondary effort and flow. More concisely, for a lossless/ideal gyrator, the primary power is equal to the secondary power. Practical examples of gyration cross energy domain boundaries; thus, providing the means to describe systems of systems interdependencies in a form which can be conveyed to a computer.

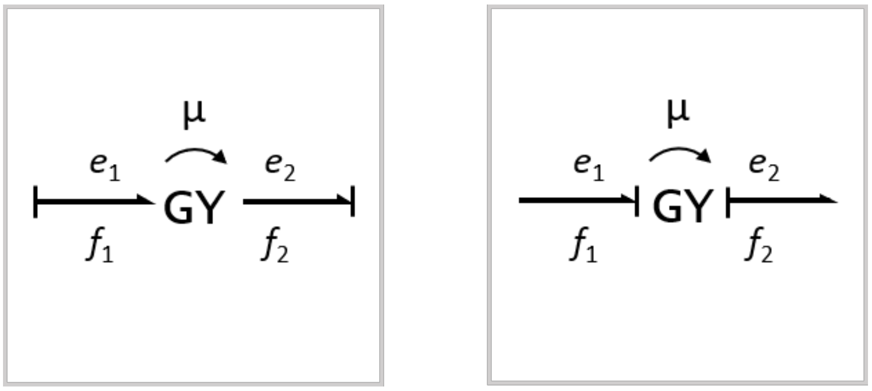


Figure 4.8. Ideal Gyrator with Causality

One such example is the arbitrary (ideal/lossless) DC electro-roto-mechanical machine — a machine which crosses the energy domain boundaries of electricity and rotational kinetics. In such a machine — a brush DC motor for example — output torque is a function of the product of a gyration ratio and an electrical current, while the back EMF in the electrical domain is a function of the product of the armature rotational velocity and the same gyration ratio. For any gyrator element, the bond graph representation includes two bonds about the element, each carrying primary and secondary effort and flow information respectively and an indicating symbol carrying the transformation ratio about the element.

The issue of causality, alluded to above as a consideration in power directionality, is now considered directly and illustrated in Figure 4.8. In bond graph theory, causality captures the essence of information responsibility throughout a system. Each element in a system is responsible for the determination of either the flow or the effort information about itself — i.e., the effect. Further, the information for which the element is not responsible

— i.e., the cause, is determined by the rest of the system. A cause-and-effect relationship must be defined for every system element and such relationship/character is captured by a single bond.

Referring back to the inertial element. Described above, the flow-effort character is such that flow is the effect (a function) of the complete history of the effort information — the cause — about the element. Thus, the inertial element is responsible for the flow about the bond and that effort, applied to an inertial element, causes the effect of flow about the same element. Similar causality holds for the compliance element. Namely, for a compliant element, effort is caused by (is a function of) the product of a function (or constant) and the history of flow applied to the element. Given this, the compliant element is responsible for the effort about the bond and that flow applied to a compliant element causes the effect of effort about the same element. By convention, this causality is denoted by an orthogonal stroke, a causal stroke, located at the edge of the bond in which effort is causal [52, 53]. For the inertial element, the causal stroke appears at the inertial element side of the bond, indicating that system effort applied to the inertial element is the cause of flow about the inertial element. Similarly, the causal stroke of a compliant element bond appears away from the compliant element side of the bond, indicating that effort about the compliant element and applied to the system is the effect of flow applied to the compliant element caused by the system. By the same reasoning, the causal stroke appears on the system side of the effort source, and the element side of the flow source. A resistive element bond has the character such that the bond may cause one of either the flow or the effort, but never simultaneously both. The bond applies control to the system dynamically based on its connectivity to the system [52]. Where flow information is the cause applied to the resistive element, effort information will be the effect about it. Conversely, where effort information is the cause applied to the resistive element, flow information will be the effect about it.

Causality must be established for each bond of the of the multi-bond elements, Following the reasoning established above, where the causal stroke of a transformer element primary bond appears on the system side, the causal stroke of the secondary bond appears on the transformer side, and vice-versa. In other words, where a primary system applies effort to a transformer, the transformer applies causes effort about a secondary system, having the effect of flow from the secondary system about the transformer which in turn has the effect of flow about the primary system, and vice-versa. Similarly, where the causal stroke of a gyrator element appears on the system side, the causal stroke of the secondary bond appears on the system side and vice-versa. In other words, where a primary system applies effort across a gyrator, the gyrator will induce flow about a secondary system; which, in effect, will apply effort back into the gyrator, in-turn effecting the flow about the primary system, and vice-versa. This is shown in Figure 4.8.

The advantage of the bond graph language and notation is that translation of a dynamical energy system into a bond graph model, including a multi-domain energy system model — i.e. that which constitutes a flow system inter-dependency — is algorithmic. Reason follows that derivation of the differential equations which constrain the system are similarly algorithmic and thus conducive to delegation to a computer system. Intuitively, such system may then also serve to evaluate said differential equations as a critical computation in evaluation of the model overall.

4.2. PORT-HAMILTONIAN SYSTEMS

Port-Hamiltonian systems theory is a multi-physics framework intended to provide a means of system control. The theory is based on considering energy and power as the common language across physical domains and combines principals of Hamiltonian dynamics with network architecture. By identification of underlying physical structures in mathematical models, powerful insights are made available for simulation, analysis, and control [54].

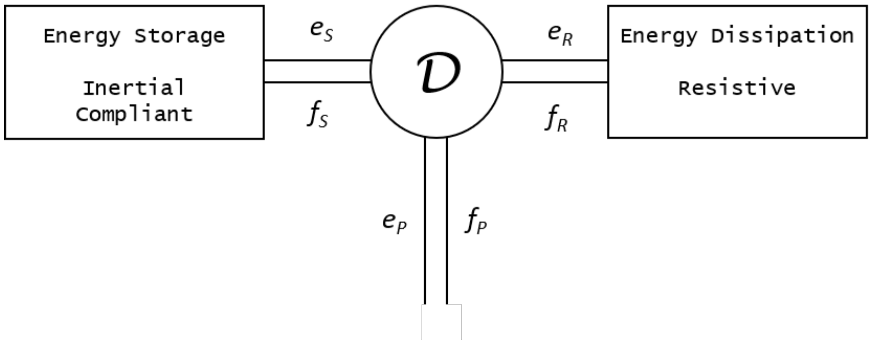


Figure 4.9. Structure of a Port-Hamiltonian System

This framework proposes that any physical system can be modeled in the form of storage elements and dissipation elements linked by paired variables/properties termed effort and flow. Effort-Flow pairs are additionally considered as power conjugates, such that the product of a paired set of effort and flow is equal to power — energy over a unit time. These power conjugates link the power of storage elements to that of dissipation elements by means of a Dirac structure which additionally functions to link power conjugates from external sources [54]. Port-based modeling considers a physical system as a set of ideal basic elements interconnected or linked by energy flow. Linking is established by conjugate vector pairs of flow and effort with a product of power. This character forms the foundational structure of the general Port-Hamiltonian System, as shown in Figure 4.9.

Hamiltonian dynamics motivate Port-Hamiltonian representations of energy elements.

$$\dot{x} = -f \qquad\qquad e = \frac{\partial H}{\partial x}(x) \qquad\qquad \frac{\mathrm{d}}{\mathrm{d}t}H = e^T f \tag{4.6}$$

In the most complex case — that of energy storing elements — effort is proposed as a function of the partial derivative of the Hamiltonian (the energy function — H) with respect to a state vector (x). For energy storing elements, flow is said to be the negative of the first order derivative of the state vector. Hence, the derivative of the Hamilton, with respect to time is the vector product of effort (transpose) and flow, or simply power.

$$R(f,e) = 0 \qquad\qquad e^T f \leq 0$$

Similarly, energy-dissipating elements are described by a linear relation of the power conjugates such that the vector product of flows and efforts (transpose) is less than or equal to zero. Energy routing elements describe a vector pair of power conjugate pairs.

$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad\qquad e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \tag{4.7}$$

$$f_1 = Mf_2 \qquad\qquad e_2 = -M^T e_1 \tag{4.8}$$

$$f = Je \qquad\qquad J = -J^T \tag{4.9}$$

The generalized transformative element, or simply transformer, links primary and secondary flow by a transformation matrix, and the secondary and primary effort by the negative transformation matrix transpose. Generalized gyrators map efforts to flows such that a flow vector is equal to the product of a gyration matrix (J) and an effort vector where the gyration vector is skew symmetric — equal to the negative of its transpose.

$$e_1 = e_2 = ... = e_k \qquad f = f_1 + f_2 + ... + f_k = 0 \tag{4.10}$$

or

$$f_1 = f_2 = ... = f_k \qquad e = e_1 + e_2 + ... + e_k = 0 \tag{4.11}$$

Finally, a set of ideal interconnection and constraint functions state that either, for a given interconnection, all efforts are equal and flows summative, or all flows are equal and efforts summative [54].

$$e^T f = e_1 f_1 + e_2 f_2 + ... + e_k f_k = 0 \tag{4.12}$$

The rules on energy routing elements and interconnections and constraints are such that power conservation holds for the power conjugate pair vector space and that a number of (independent) linear equations can be produced equal the power conjugate vectors' shared dimension. This set of elements, interconnections, and constraints is grouped into a geometric object, termed a Dirac structure.

This element-based architecture and language offers powerful tools to synthesize the behavior of physical systems[54] in similar form to that of bond graph theory. Similarities are evident between the two frameworks from terminology to rule sets. While Port-Hamiltonian approaches group the behavior of energy storing elements such as inertial and compliant elements and represent resistive elements as energy dissipative, these representations are functionally similar in that singular effort-flow port pairs unite energy storage and dissipation into routing. Both frameworks include multi-port/multi-bond representations of inter/intra-domain energy transformation elements — the transformer/gyrator, respectively. Across these frameworks, the shared terminology and structure motivates much of the terminology used to describe the FLASH interfaces in Chapter 5 which must be captured by an implementing software engineer.

5. FLOW AGENT-SYSTEM HYBRID (FLASH) FRAMEWORK

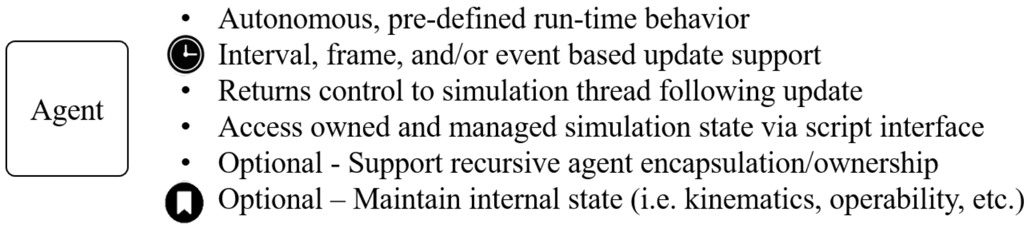
The governing constraints and behaviors observable about a given flow system from the lens of a single domain (i.e. energy, transportation, batch commodity, generally any resource), is well-established. Several such flow systems are described in previous sections. The objective of the FLASH framework is to unite these intra-domain systems through common points of interface such that singular flow system models remain functional in isolation, while additionally describing the means by which large, complex, and interdependent systems may be additionally modeled in a scalable form with the primary objective of advancing the standards of quality in flow system modeling across industries. As discussed, industry conventions include domain-specific models which are high-fidelity, but which assume the driving input of sources to always be readily available such that nominal efforts/flows are given.

Provision of a uniform interface through-which flow systems of any domain may be defined will allow software engineers to support analysts across industries to employ flow system specific knowledge to efficiently reproduce domain models. In doing so, software engineers will become familiar with the requirements to deploy a system of similar complexity in the AFSM simulation space on which the FLASH is necessarily built. This will reduce the need among analysts to use multiple software tools to produce any singular domain flow system. Instead, a single tool suite with a uniform and script-accessible state interface not only yields recognizable domain results, but also allows for interaction across domain boundaries at simulation run-time. Building such a unified space requires a limited number of utilities which are partially available through use of particular industry tools to be described in later sections. The AFSM requisites needed to support this FLASH framework and derived types, though presently fragmented across tools across industries, are implemented in large part already and so highlighted ins such cases, but are not described to the level of implementation in this body of work.

5.1. FOUNDATION

The FLASH framework bears a set of underlying requirements. These requirements may be met by a tool or set of tools such as computer software already written or to be written and are referred to here as AFSMs. To support flow unity, the AFSM must support scenario definition. A scenario describes objects and systems, and the user-specified rules which govern such objects (entities and agents) in the context of AFSM simulation space. To improve scalability, the underlying AFSM should support scenario definition within a geo-spatial context such that agents maintain some spatial position within the relative frame of a celestial body such as Earth, including a latitude, longitude, and elevation/altitude. Alternatively, for space-based/orbital agents, an AFSM should support scenario-based orbit definition. Pairing a spatial context (i.e., location) with each agent in the scenario space then serves as a boon to iterative design and analysis when combined with spatial visualization. While not strictly required, this significantly aids in the extension of existing flow systems and is in some form supported for each industry tool considered in Chapter 6.

5.1.1. Agent Architecture. A core AFSM requirement is support for scenario definition of agents which are, at run-time, functionally autonomous — behaving based on a set of scripts which the user defines. Such agents may recursively encapsulate sub-agents such as (components, parts, etc.) which behave similarly within the boundaries of their scope. At the scenario/global scope, agents are named unambiguously and must support the ability to temporally ‘update’ at an agent defined interval such that at every simulation run-time update the agent is able to execute a pre-defined set of behaviors which consider scope-bounded simulation state and use particular state information to alter internal agent state. In the case of sub-agent containerization, agents then invoke updates on attached components with compartmentalized responsibility and scope. Update completion must return control to the simulation environment to maintain run-time scheduling constraints, maintaining that all simulation entities are provisioned computational assets to complete scheduled updates. Agents must support run-time invocation of script interface methods via script contexts. Said requisites are summarized in Figure 5.1. A sub-agent UML architecture is then presented in Figure 5.2



- Autonomous, pre-defined run-time behavior
- Interval, frame, and/or event based update support
- Returns control to simulation thread following update
- Access owned and managed simulation state via script interface
- Optional - Support recursive agent encapsulation/ownership
- Optional – Maintain internal state (i.e. kinematics, operability, etc.)

Figure 5.1. Agent Requirements

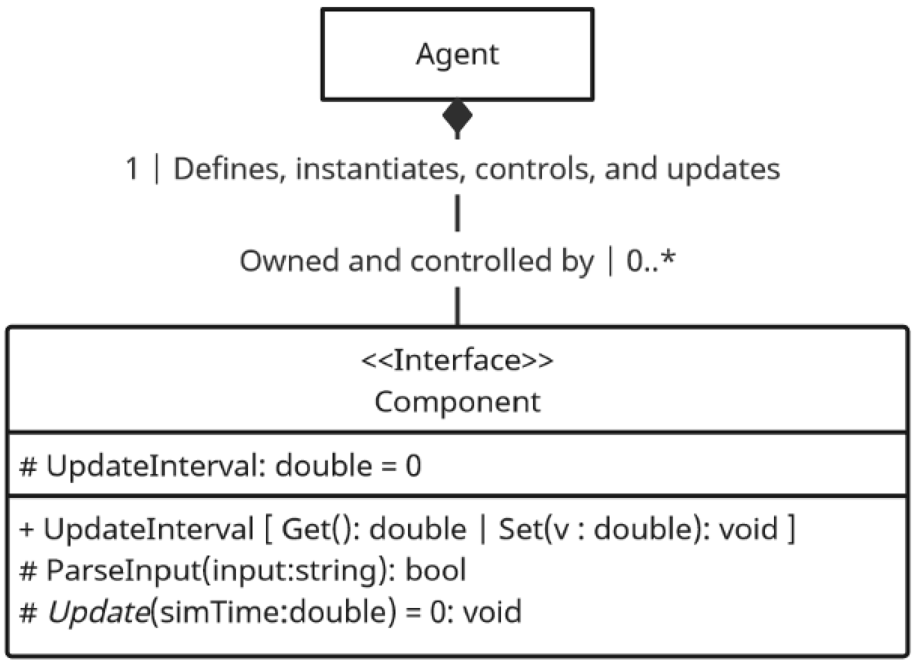
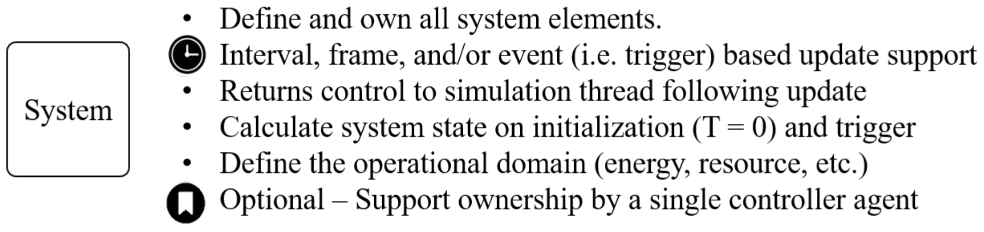

Agent
1 | Defines, instantiates, controls, and updates
Owned and controlled by | 0..*
<<Interface>>
Component
# UpdateInterval: double = 0
+ UpdateInterval [Get(): double | Set(v : double): void]
# ParseInput(input:string): bool
# *Update*(simTime:double) = 0: void

Figure 5.2. Agent/Sub-Agent Architecture UML

5.1.2. Flow System Architecture. The flow system aspect of the FLASH architecture includes two distinct parts — system scenario definition and system simulation architecture. The following section describes a duality about the individual elements of a particular flow system. The system must function as both a sum of parts and as a whole. Therefore, a template of the pattern of ownership and management between systems and agents is needed.

5.1.3. Scenario Definition. Even single domain flow systems are often complex and constrained by holistic physical dynamics, Therefore to achieve scalability, the AFSM must support definition of a single localized and unified system definition within the scenario. Mono-domain REFS systems are defined discretely and are therefore containerized. This containerization directly enables portability across frameworks. Within a single system definition, all intra-domain system elements are to be defined. Fundamental element types are described to follow, but for the purposes of system definition, suffice to state that the system encapsulates (and owns) the individual definitions of each of its discrete elements. Included in the system definition is the operational/energy domain in which the system operates as well as a system-level update trigger such as an interval or event. Such triggers affect run-time behavior, and are described in detail to follow. For compatibility with an agent-based element-referential architecture, a unique identification pattern is necessary to support unambiguous references to system element definitions from particular managing agents/sub-agents. Fundamental responsibilities of the system are denoted below in Figure 5.3.



- Define and own all system elements.
- 🕒 Interval, frame, and/or event (i.e. trigger) based update support
- Returns control to simulation thread following update
- Calculate system state on initialization (T = 0) and trigger
- Define the operational domain (energy, resource, etc.)
- 🔖 Optional – Support ownership by a single controller agent

Figure 5.3. System Specifications

By scenario-defined reference (and depicted in Figure 5.4), a given agent may claim management of zero or more elements of: a single system, multiple discrete systems sharing a common domain, and/or multiple systems spanning multiple domains. A generalized form of this relation is shown in Figure 5.5. This claim of state responsibility by an agent unifies defined elements under one controlling manager (agent/sub-agent).In combination with scripting, this affords end users the power to define arbitrary flow-system element interactions and interdependencies. This lends itself to multi-resolution modeling, an oft desired feature which is notably inaccessible under strict FUMES patterns.

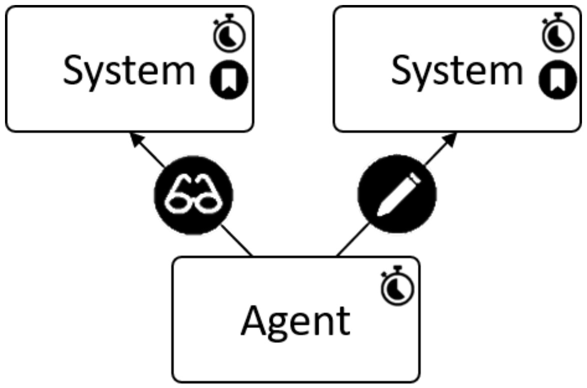


Figure 5.4. High-Level Agent-System Interaction

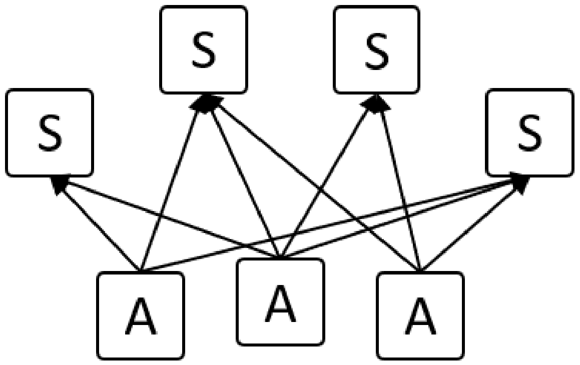


Figure 5.5. Agent-System Multi-Interaction

5.1.4. Simulation Architecture. An AFSM must support agent updates at an agent-defined interval. Notably this behavior will hold in principle as well for a system, which must be definable at the scenario level (global), but may also be definable at an agent level as a component/sub-agent attachment to a particular agent. In either case, the system must be named unambiguously within its scope. In the latter case, a user provision and/or default definition of update triggers is not strictly necessary, If no trigger is defined, the simulation must minimally invoke a system run-time update at initialization. For any simulation of non-trivial duration, a default trigger is additionally necessary. Depicted in

Figure 5.6, guaranteed triggers function such that changes in system structure (new and removed connections), in addition to state changes in particular elements. may be used as input for iterative recalculation of system dynamics/simulated state.

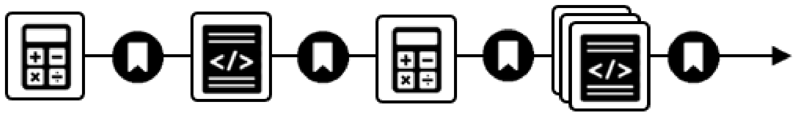


Figure 5.6. Simple Agent-System Turn Cycle

Regarding non-periodic triggers, some implementations — particularly in such AFSMs where a high parallel processing bandwidth and safe threading architecture is in effect — may accept that a system defined update interval is not necessary, but rather the individual updates — particularly changes in state — for any given system element, may signal a system-level update. However, such an implementation lends itself to run-time thrashing where proper safeguards (such as cooldowns) are not in place to prevent (or properly limit) cyclic update triggers between a system and its elements through agent managers.

5.1.5. Elemental Reference. Within the simulation space, each system is considered jointly both as an operational whole and as a discrete set of elements managed by agents and/or sub-agents — i.e., components/parts. To rectify this duality, a scenario must define system elements within the scope of a system block. Each element must be named unambiguously within the scope of the containing system. Unique naming then allows for a particular system-defined element to be referenced within the definition of an element manager. A given element manager (agent) behavior is defined within the scope of a single agent which may include a variety of a system-defined elements.

Thus, the proposed referential structure, depicted in Figure 5.7, combines the encapsulation benefits of a self-contained scenario definition of a system, lending itself to use in describing large domain systems. Combining this modularity with the advantages of trigger-based, agent-defined interactions and behaviors, multi-resolution cross-domain interdependencies may be modeled.

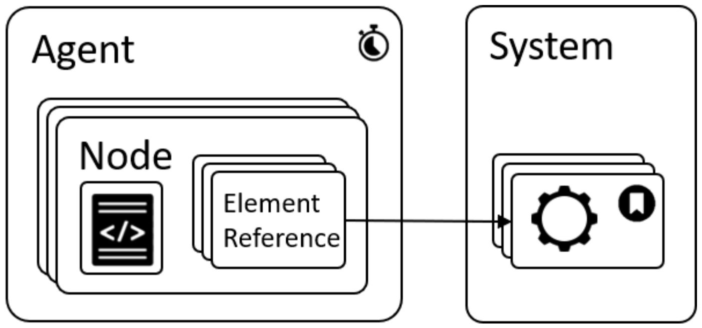


Figure 5.7. Referential Agent Management

5.1.6. Script Interface and Context. The script interface is provided by the AFSM and extended by the FLASH framework implementation. System elements, captured as the responsibility of a managing agent, must provide a uniform, domain independent, interface which is accessible within a script context. Both interval and signal-based behavior of a given agent (and its sub-agents/components/parts) are support invocation of a script context.

Script Context

- Direct invocation by external triggers
- Read immutable power components (i.e. effort, flow), constraints, etc.
- (Re) Calculate resultant changes for managed system elements
- Write updated state information to mutable referenced elements
- Support external dynamics integration (i.e. control, cyber, weather)

Figure 5.8. General Script Context Specifications

This context — usually programmatic in nature — must support extension for automatic inclusion, access, and in some cases mutation, of referenced element data/state within owning parent systems. With this requirement met, domain interdependencies may be described for a given scenario to support emergence in cross-domain interaction at simulation run-time. Figure 5.8 enumerates the fundamental minimum specifications of a script contact. Such specifications are generally applicable to system, agent, and out-of-scope interactions of script-enabled-capabilities.

5.2. APPLICATION

To solidify understanding a simple, but non-trivial example is presented which highlights the necessity of modeling multi-domain critical infrastructure inter-dependencies. Simultaneously, the example demonstrates the required scenario architecture and simulation run-time behavior of the implemented FLASH Framework.

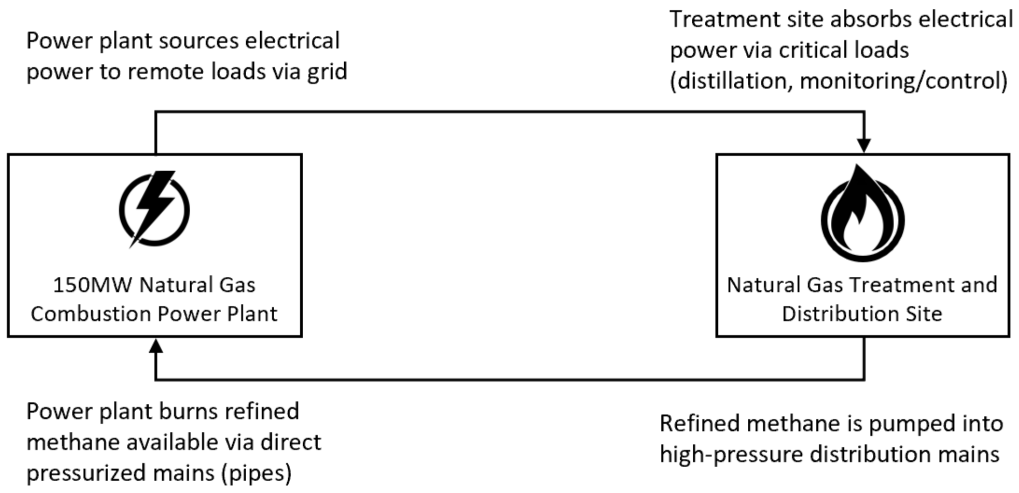

Power plant sources electrical power to remote loads via grid
Treatment site absorbs electrical power via critical loads (distillation, monitoring/control)
150MW Natural Gas Combustion Power Plant
Natural Gas Treatment and Distribution Site
Power plant burns refined methane available via direct pressurized mains (pipes)
Refined methane is pumped into high-pressure distribution mains

Figure 5.9. Scenario Setup and System/Agent Definition

5.2.1. Example.

In this simple example, a direct circular interdependency is defined between two components of modern critical infrastructure: A natural gas combustion turbine power plant and a high capacity natural gas treatment and distribution facility (NGTF). As depicted in Figure 5.9, the power plant functions as a sink/consumer of methane gas, which directly fuels the kinetic rotation of a turbine and mechanically connected electro-magnetic generator. Said generator, when operational, energizes the electrical grid, as described in Figure 5.10.

Similarly, Figure 5.11 describes the function of the NGTF.

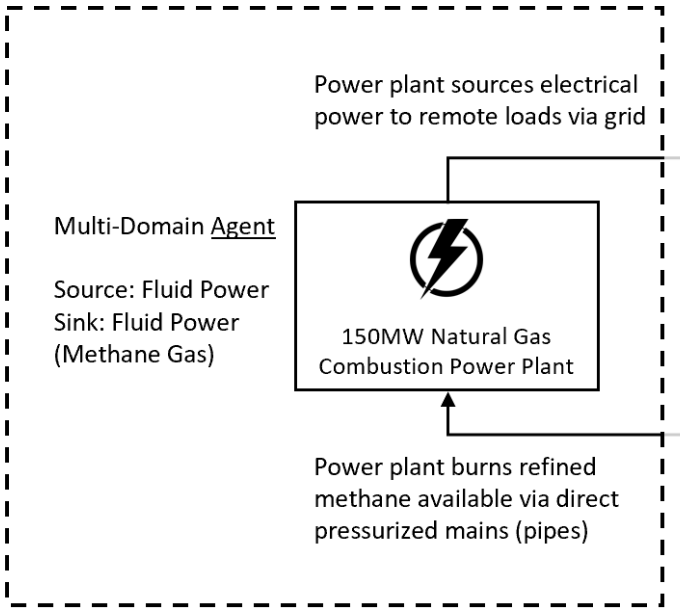

Power plant sources electrical power to remote loads via grid
Multi-Domain Agent
Source: Fluid Power
Sink: Fluid Power
(Methane Gas)
150MW Natural Gas Combustion Power Plant
Power plant burns refined methane available via direct pressurized mains (pipes)

Figure 5.10. Multi-Domain Power Plant Agent

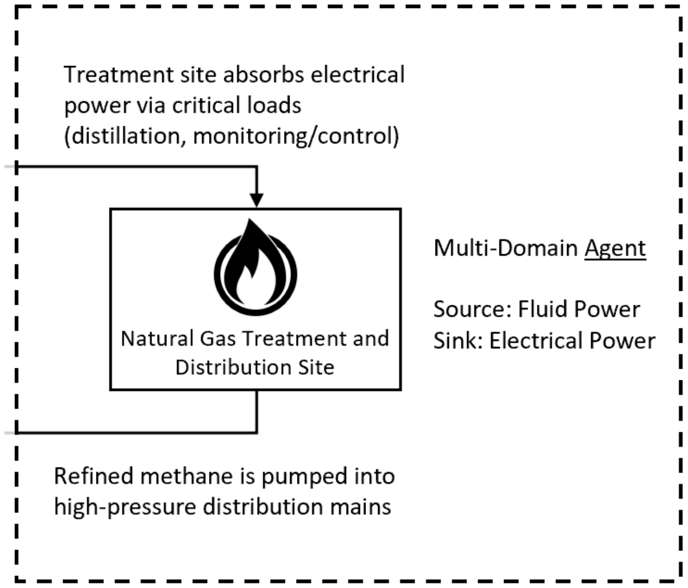

Treatment site absorbs electrical power via critical loads (distillation, monitoring/control)
Natural Gas Treatment and Distribution Site
Multi-Domain Agent
Source: Fluid Power
Sink: Electrical Power
Refined methane is pumped into high-pressure distribution mains

Figure 5.11. Multi-Domain Natural Gas Site Agent

In this case, all gas is supplied exclusively by the NGTF, while all electrical power on the grid is provided by the functioning power plant. The NGTF consumes electrical power as accessed via supporting grid infrastructure. This power is consumed at the NGTF site, powering control and monitoring electronics, facilities, communications, distillation boilers, and other related sub-infrastructures.

A clear interdependency is evident. That is to say, for this well defined universe, if one of these becomes non-operational (arbitrary cause) it should be evident based on the tight scope of the scenario description that the other agent should should also become non-operational within a short time frame (seconds to minutes). This lends itself to a host of questions. The first: If they require each other to function, how did either one start working in the first place? The answer is clear given consideration of Figure 5.12.

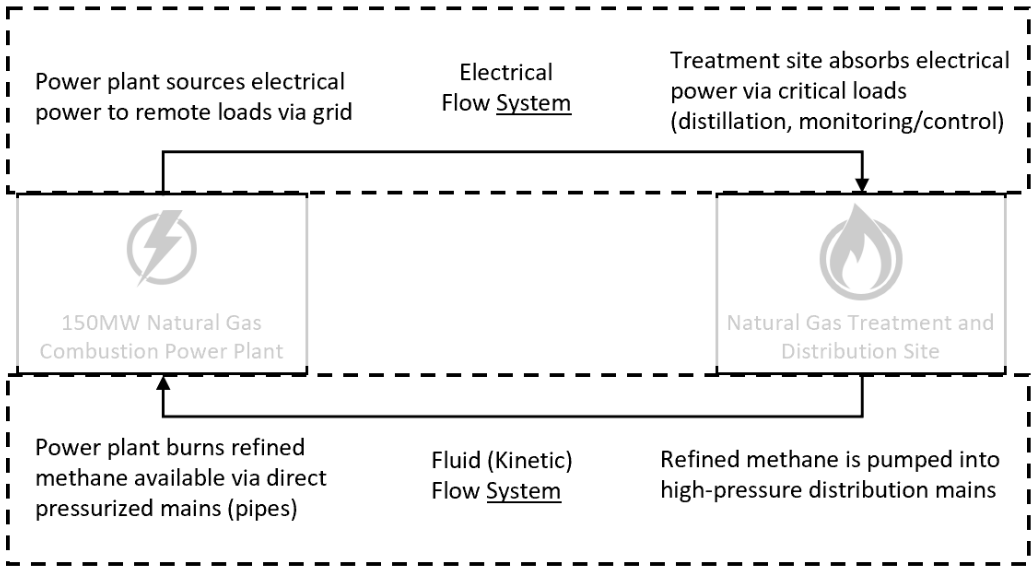

Power plant sources electrical power to remote loads via grid
Electrical Flow System
Treatment site absorbs electrical power via critical loads (distillation, monitoring/control)
150MW Natural Gas Combustion Power Plant
Natural Gas Treatment and Distribution Site
Power plant burns refined methane available via direct pressurized mains (pipes)
Fluid (Kinetic) Flow System
Refined methane is pumped into high-pressure distribution mains

Figure 5.12. Define Systems within Energy/Resource Domain Boundary

Figure 5.12 highlights the scenario-defined architecture which permits these systems to start - and continue to function - without one another. It is clear from the illustration that neither system is directly dependent on the other. Therefore, each system must define it's initial nominal state. This is achieved by flow calculation. In each domain, standard flow analysis techniques are recognized by professionals within the domain. As discussed in Chapter 2, Newton-Raphson is an industry standard for the calculation of electrical power flow about a system. Similarly, Chapter 2 identifies Hardy Cross as the preferred flow analysis technique across the fluid domain space. These algorithms are those which must be used to calculate the initial flow about each system in order for an implementation of the FLASH framework to be complete. This requirement includes an underlying requirement that each system must provide a minimally viable amount of data with which the complete system state may be calculated. Such problems are abundant for every domain of interest. Each system in this case is trivially simple (each having a single source and a single sink only per the specification in Figure 5.9), thus such algorithms are not strictly necessary in this trivial case, but must be executed none-the-less. Therefore, for every discrete system to be simulated, a nominal state is calculated once at the time of simulation initialization without a requirement to consider interdependencies. This operation is the focal of Figure 5.13.

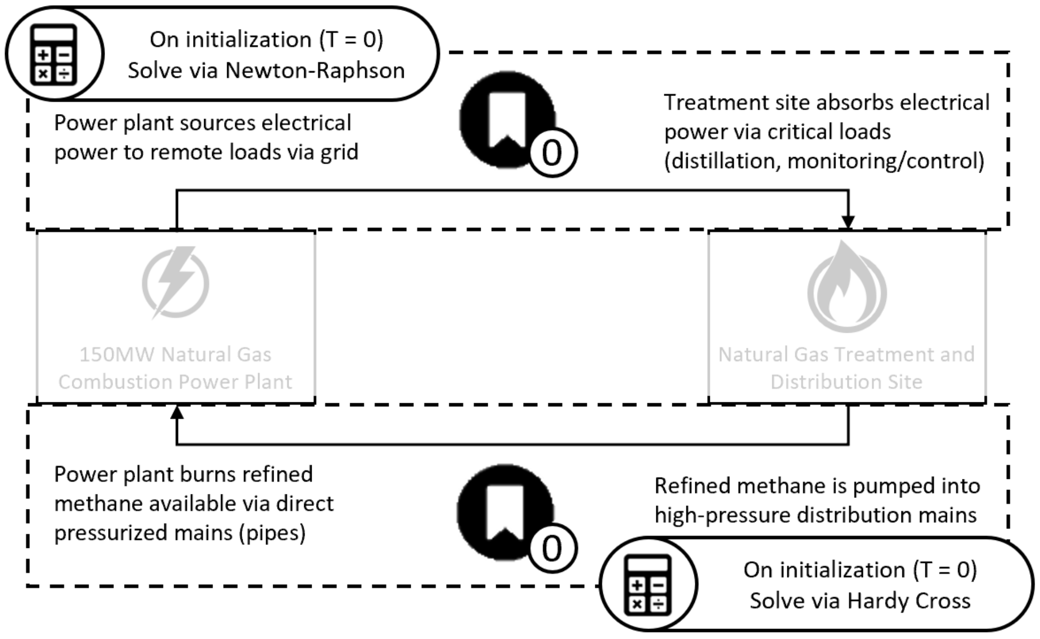

On initialization (T = 0)
Solve via Newton-Raphson
Power plant sources electrical power to remote loads via grid
0
Treatment site absorbs electrical power via critical loads (distillation, monitoring/control)
150MW Natural Gas Combustion Power Plant
Natural Gas Treatment and Distribution Site
Power plant burns refined methane available via direct pressurized mains (pipes)
0
Refined methane is pumped into high-pressure distribution mains
On initialization (T = 0)
Solve via Hardy Cross

Figure 5.13. Simulation Initialization Behavior (T = 0) - Calculate Nominal System State

Following initialization, on agent-defined update triggers (usually time-interval based), agents may observe and mutate particular values of particular system elements based on the scenario defined management designations. In this case. The power plant is a read manager of the sink element of the natural gas system, and is a read/write manager of the source element of the electrical system. In this way, the power plant agent may, at any time following initialization, read the values of each managed node and execute conditional logic and flow control (in a user defined script context) based on those values. Such conditional logic is likely — but not strictly required — to include possible write operations against read/write managed nodes, and may additionally affect other arbitrary (out of scope) agent and agent managed properties per the values read. This event/frame/interval triggered agent update enables run-time inter-dependency coupling and is the focus of Figure 5.14.

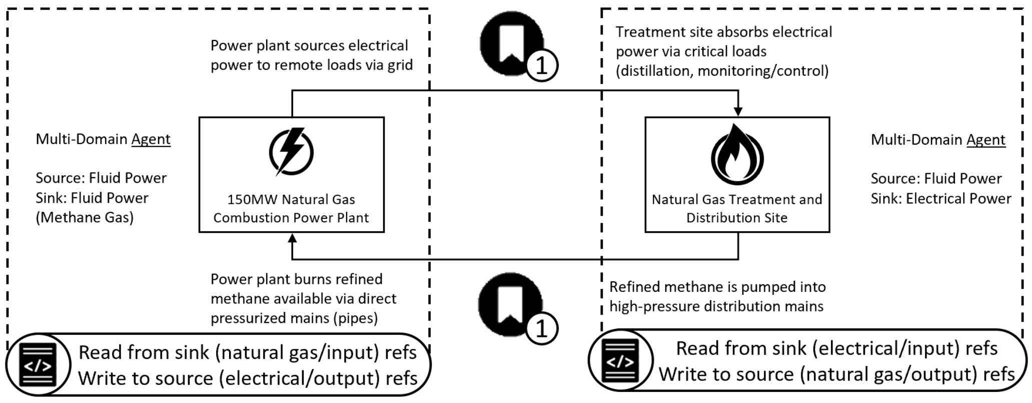

Power plant sources electrical power to remote loads via grid
Treatment site absorbs electrical power via critical loads (distillation, monitoring/control)
Multi-Domain Agent
Source: Fluid Power
Sink: Fluid Power (Methane Gas)
150MW Natural Gas Combustion Power Plant
Natural Gas Treatment and Distribution Site
Multi-Domain Agent
Source: Fluid Power
Sink: Electrical Power
Power plant burns refined methane available via direct pressurized mains (pipes)
Refined methane is pumped into high-pressure distribution mains
1
1
Read from sink (natural gas/input) refs
Write to source (electrical/output) refs
Read from sink (electrical/input) refs
Write to source (natural gas/output) refs

Figure 5.14. Agent Updates - Read Inputs (Sinks), adjust Outputs (Sources) Accordingly

Following agent updates, system triggers may yield system-level updates to include the re-calculation of system state. In the case that managing agents have modified independent flow variables such as increasing load values, modifying transformer values — having the most acute impact — reducing a source output to 0. In this case effect propagation is exponential, having the possibility to negatively impact the operation of any and all interconnected infrastructures. In the universe described by this example, the effect propagation is already known. Thus, the process follows to reach the known outcome.

5.2.2. Execution. Assume nominal, stable, inter-dependency coupled operation between the power plant and NGTF, up through time $t - 1$. At t, assume that a malicious cyber attack renders the NGTF non-operational because a virus infects the control software and closes all valves connecting NGTF outflow to the natural gas mains described previously, while simultaneously opening emergency valves resulting in rapid de-pressurization of the mains. Thus, the agent writes a new value to the managed source element within the natural gas flow system instance. To explore the full problem space, the power plant agent also updates at t. Although the attack is in effect at the NGTF, the effects will not propagate until a system recalculate until a system recalculation occurs.

It follows then, at $t + 1$, the natural gas flow system updates. Given the source head (pressure) value is rapidly decreasing (if not already zero), pressure at the sink side (power plant side) of the natural gas system decreases (assume negligible friction/distance). Assuming multi-threaded support, the system will issue a mutual exclusion lock over all owned elements prior to solving flow with Hardy Cross. In this case, and in the case of a single-threaded simulation, no simultaneous updates to the natural gas system are possible at $t + 1$. When system state calculation resolves, the natural gas system releases locks on owned elements' data. Rapid depressurization leaves the natural gas system at a uniform zero relative pressure (1 ATM) at t. Thus, Hardy Cross yield's a zero flow about every pipe in the network on system state recalculation at $t + 1$. Assume the NGTF agent includes cyber-security mechanisms, and that an emergency reboot will clear any ongoing attack effect at

The rapid propagation of this attack will foil recovery as well. To understand this, it is necessary now to follow the effect chain on an order of magnitude to highlight the severity and possible attack impact in a less-trivial system:

- O^{th} order - $t + 0$ - Cyber attack occurs at NGTF. Pipe mains depressurized.
- 1^{st} order - $t + 1$ - Natural Gas system state recalculated. All pressure is relative zero. It follows that all flow is zero.
- 2^{nd} order - $t + 2$ - Power Plant Agent update trigger - Scripts invoked, natural gas sink element power = pressure * flow = 0. Scripted logic includes interdependency linkage. (i.e. If natural gas input power is less than min power required, force generator outage and disconnect from grid at switch yard).
- 3^{rd} order - $t + 3$ - electrical system instance update triggers - source voltage = ground (zero). Without proper protective infrastructure (none defined here), severe back transience from the NGTF propagates through the grid. - Higher complexity than is conducive to this example - Instead, assume protective breakers, fuses, surge

protectors, activate ideally and all grid node voltages are immediately equal to ground. With zero potential about all nodes in the system, Newton-Raphson necessarily yields 0 flow across all reference points - A 100% grid blackout.

- 4^{th} order - $t + 4$ - NGTF update triggered. Interdependency script logic executes. NGTF read managed electrical system sink values read power = voltage * current = 0. Primary electrical system failure at NGTF at $t + 4$. No backup electrical sources are defined. All electrical systems immediately loose power. NGTF agent ceases operation. Power plant agent also updates at $t + 4$. No grid power. No reserve power defined. Power plant agent ceases operation.

The effect chain above shows catastrophic critical infrastructure systems-of-systems failure in only 4 time steps. This effect chain is relatively linear due to the highly restricted scope of the scenario. In practice however, no such simplicity is found. Instead, it is apparent even here that exponential effect chains may propagate about a highly inter-dependent simulated system-of-systems space ($O(n^m)$). Thus, it is conceivable that such devastating propagation may emerge in real world critical infrastructures. Not only is it conceivable, instances of the same occur at high frequency [4, 11, 17].

The science and technology which has enabled rapid societal advancement and growth must be adaptable, scalable, and portable in order to enable the analysis, control, and design of the ever-more inter-dependent systems they yield. The FLASH Framework is one such adaptation. The FLASH framework is designed to simplify and modularize the problem space to enable mass adoption. It is extensible into the general flow system space to enable adaption at relatively low cost into any future emergent flow system problem set.

5.3. SYNTHESIS INTO IMPLEMENTATION

The extensibility and scalability of the FLASH framework declared previously. In support of this declaration, a detailed, software-centric, description of the framework and it's prerequisites follows.

The principles and patterns inspired by FUMES are present within the FLASH framework exclusively from the perspective an implementing software engineer. Implemented as defined here, FLASH code is unified and supports arbitrary future extension. Importantly, this FUMES inspired generalization is necessarily hidden — except at the script interface layer — to enable portability and industry adoption. Thus, the required abstract system representation is such that derivative (end-user facing) implementations inherit a declared, but undefined, set of methods (interfaces). The patterns which follow adhere to modern principles of object-oriented software architecture and design. Detailed descriptions and accompanying UML are presented such that the adept software engineer may translate discussion and theory into programmatics. The proposed architecture, while decomposed into individual class-based UML structures here is presented as a complete extension structure with relational linkage in 8.

5.3.1. Systems. Generalization prohibits direct system instantiation. Instead, indirect instantiation is supported through means of derivative system types which define inherited system interfaces. The abstract system type will declare and define an interface by which a particular system instance may be unambiguously identified in the containing (usually global) scope, as well as an interface by which a periodic update interval and other update triggers may be specified. System instances maintain complete knowledge and ownership over all owned elements and the interconnections among them (edges). This pattern is captured by Figure 5.15.

The system type will declare, but not define, interfaces which provide reference — or reference information necessary — in order for agent-based managers to access and mutate individual elements within a set of element-defined constraints. The type must declare,

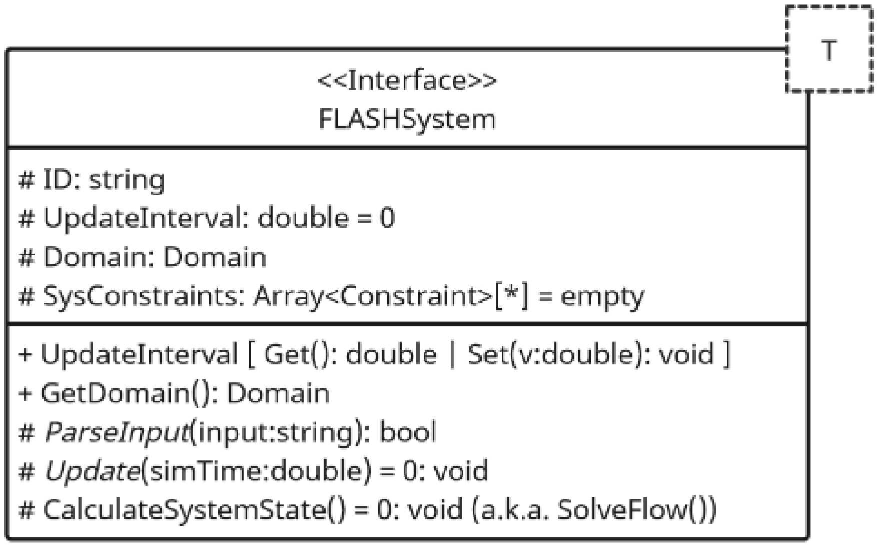

T
<<Interface>>
FLASHSystem
# ID: string
# UpdateInterval: double = 0
# Domain: Domain
# SysConstraints: Array<Constraint>[*] = empty
+ UpdateInterval [Get(): double | Set(v:double): void]
+ GetDomain(): Domain
# ParseInput(input:string): bool
# Update(simTime:double) = 0: void
# CalculateSystemState() = 0: void (a.k.a. SolveFlow())

Figure 5.15. Abstract System UML

but not define, interfaces by which the domain of a system instance may be determined, by which the system-wide power and power conjugates are to be calculated, and by which the unit type associated with effort, flow, and power are respectively measured. The scenario system definition is responsible for the specification of: the domain of operation, the domain-specific names of owned elements, the static and initial dynamic characteristics (nominal system/element behaviors), and finally the edges which link element pairs.

Declaration of domain support, is an assurance that a system model sub-framework (derivative type) is implemented and minimally defines all inherited abstract interfaces such that a system of the target domain may be instantiated in simulation. Illustrated in Figure 5.16, the system instance then invokes its particular implementation to solve power and flow at simulation initialization. State resolution algorithms may then be optionally invoked for system state recalculation via system defined triggers and/or by means of slotting power and flow recalculation against linked signals (events) occurring at simulation run-time.

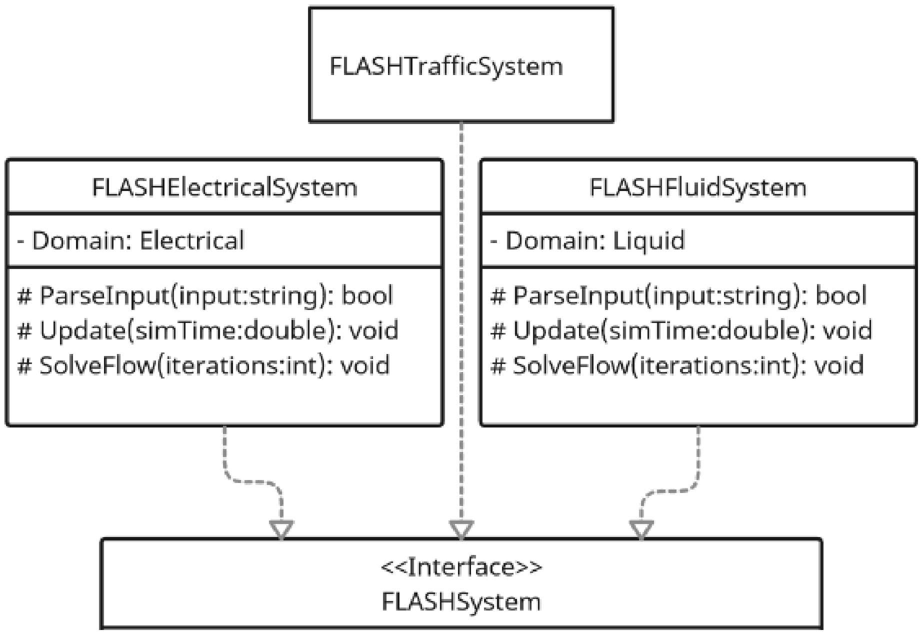

FLASHTrafficSystem
FLASHElectricalSystem
- Domain: Electrical
# ParseInput(input:string): bool
# Update(simTime:double): void
# SolveFlow(iterations:int): void
FLASHFluidSystem
- Domain: Liquid
# ParseInput(input:string): bool
# Update(simTime:double): void
# SolveFlow(iterations:int): void
<<Interface>>
FLASHSystem

Figure 5.16. Example Implementations of the System Interface

5.3.2. Elements. The FLASH framework abstracts all system elements into a graphical representation. For any domain, any given node which maintains discrete state within a system is defined as an element . Importantly though, an element is itself abstract and cannot, without the constraints and boundaries (i.e., character) of a domain, be invoked. Rather, an element is a foundation on which to build an element of a particular domain. This foundation is reflected in Figure 5.17. In conjunction with element derivatives, which for a domain may be invoked, elements are inter-linked for a particular domain by edges, detailed to follow, to form systems. An element will declare — but not define — interfaces into element derivatives, which must define the behavior of said interfaces in order to be invoked in a simulation. Such element interfaces to be declared, but not defined, are those which describe the state of relative potential (effort) at, and flow about said element. Effort is referential with respect to a system defined potential of reference. An element may (for convenience only) declare or declare and define a method by which power is characterized by the power conjugate product.

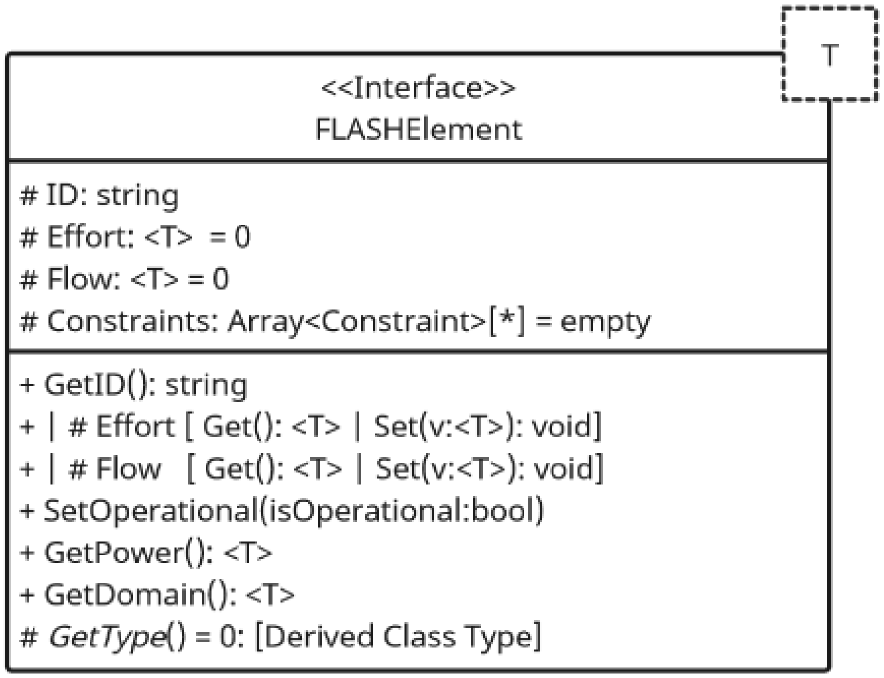

T
<<Interface>>
FLASHElement
# ID: string
# Effort: <T> = 0
# Flow: <T> = 0
# Constraints: Array<Constraint>[*] = empty
+ GetID(): string
+ | # Effort [Get(): <T> | Set(v:<T>): void]
+ | # Flow [Get(): <T> | Set(v:<T>): void]
+ SetOperational(isOperational:bool)
+ GetPower(): <T>
+ GetDomain(): <T>
# GetType() = 0: [Derived Class Type]

Figure 5.17. Abstract System Element UML

The element must additionally declare and may optionally (preferably) define a means of unambiguous identification within its containing scope. The initial state of the element is to be described by the system either by an end-user interface into the scenario or by default behavior specific to the derivative domain. The element must declare (or support a script-based) interface into an unambiguous description of nodal function, wherein definition is the responsibility of the derivative. An element must be polymorphic. In other words, the general element must support type derivation into support of arbitrary data types. Element derivatives, typically with data types varying by domain, will morph the element to the desired/necessary form as a requisite to invocation/instantiation. The initial power state of an element is to be described by a containing system. Varying by derivative type, this responsibility includes declaration, but not definition, of constraint and requirement interfaces on, target, and/or initial power conjugate value/state and power itself (henceforth, power or state information). Additional constraints may apply to power information derivatives (rates of change) of order N. For an abstract domain derivative to be further derived into an invokable type, power information value and derived value constraints may be declared or declared and defined, but such structure is not implemented within this work body. An element during instantiation/construction will provide reference by unambiguous identification or direct memory addressing, to the owning system. An element cannot exist without nor outside of a valid parent system, and element interfaces described here must be publicly accessible within the program space.

With all abstract interfaces defined, the truth of the element state is observably the responsibility of the element derivative type instance. An element, by name and with the use of supporting scenario/simulation infrastructure, is referable by agent components, which may pre-declare power information data types and domain to enforce programmatic compatibility in the scenario space or inherit typing and component domain by a valid link. A flow system nodal reference component (i.e. flow component) may exist in the scenario space without reference to a valid component, but must make a valid reference to a system element definition prior to simulation run-time, else having no operable effect on the simulation and requiring guards when component script interface methods would otherwise be invoked, minimally warranting a warning (more often an error) at simulation initialization/startup. An element must, declare an interface into the referencing agent flow component(s) in (preferably) in consistent form to the interface provided for parent system reference. The system element must provide (minimally) value (constant state) access to a serial container of N references to directly connected, system-defined edges. An agent flow component is expected to declare or declare and define a complete script API to facilitate access and mutation of nodal power state information. Element derivatives then limit or propagate control into appropriate interfaces by deletion or (override) definition of abstract element interfaces — i.e., effort sources provide an effort information mutator and accessor, but only flow information accessor. Element interfaces intended to provide data access will provide such access by return of value or constant reference and in either case declare, usually by keyword, the constant nature of the interface on the containing type. This constraint will additionally propagate through a script provided by an agent flow component.

A whole number N many agent power/energy components/sub-agents may be managed by a particular agent and update at independent and/or uniform intervals, defaulting to the top-level agent update interval. Race conditions on shared overlap times in multi-threaded and single-threaded simulation environments are to be handled deterministically (consistent ordering) as implemented by supporting AFSM infrastructure to guarantee reproducible results. Within the agent and component scope update interval and script context that cross-domain power inter-dependencies may be defined.

For example, in the case of a simple agent model of an electrical power plant, an external water system sink may be linked as a flow component in addition to a component which links to an electrical generation source. In the simplest inter-dependency representation, a script interface may condition electrical effort production on the provision of a certain minimum flow of water as described by the power state of the water sink agent component. This is a lowest fidelity representation of cross-domain gyration. Building on this, subsections to follow describe domain-independent descriptions of power elements.

5.3.3. Source. A source is a fundamental element derivative. Sources are the stimulation, points-of-origin, and driving components that provide domain input into the flow system. A flow system absent of sources has no flow, nor effort. Sources observe two forms: The effort source and the flow source, with some domains (namely that of electrical energy) and model representations requiring additional variations. The source highlights the fundamental problem of flow analysis. That is, for a given element, if one constraint on power (power conjugate) is independent and known (by value or function), the other is dependent and unknown. System level analysis, which by various methods (differing across domains), lends itself to iterative convergence on a system state based on sufficient knowledge of state of independent variables and an initial guess or approximation of the values of unknowns. The source will handle (implement/define) all abstract interfaces inherited from the parent element type. An effort source will privatize direct flow mutation and publicly implement direct effort mutation. Similarly, a flow source will privatize direct effort mutation while publicly implementing direct flow mutation. The source is a single edge element, meaning one and only one edge connects a given source into the rest of a flow system. Power information mutation should emit signals indicating the affected element such that observing objects (and slotted handlers) across the simulation may timely react to meaningful changes.

Sources have the potential for constraint in both power information magnitude and derivative values, such derivative limitations are observable in ramp-rate limits on electrical generators in power plants. In order for a generator to produce less power the physical roto-kinetic armature must slow down (decrease flow), but this desired slow is constrained by the resistive influence of momentum. Further, even at the moment a decision is made to reduce generator output, many fuel systems to the generator turbines require time to change energetic state, such as in the case of a coal furnace. Reduced coal may be injected into the furnace, but the consequences of such an operational change take time to propagate through the chain of system energy domain interdependencies. A simplification of such system-of-systems interconnection is upper and lower ramp-rate limit specification at the electrical generator source definition within the electrical system. Accurate definition of such constraints has the effect of emergent top-level cross-domain load without the need for discretely modeled internals.

5.3.4. Sink. The sink compliments the source and derives similarly from the element. Although not exclusively, sinks often specify power as the primary operational constraint with nominal underlying specifications on effort, allowing for the linear determination of a flow. Alternatively, where output flow is specified, an element voltage may be calculated, and power derived. A sink implements all abstract element interfaces and functions as an endpoint for a particular flow system in a given domain. In the nominal case, the power load/demand on a system is less than that of the present generation capacity. However, in cases where demand is greater in magnitude than supply, endpoints may be partially/completely starved for the required resource, resulting in cascading system failures. An effort sink will privatize all direct flow mutation interfaces (mutators) and publicly implement effort mutators. Conversely a flow sink will privatize all direct effort mutators and publicly implement flow mutators. Like the source, the sink is a single edge element, meaning one and only one edge connects a given sink into the rest of a flow system. Power information mutation should emit signals indicating the affected element such that observing objects (and slotted handlers) across the simulation may timely react to meaningful changes.

5.3.5. Junction. The junction is a power routing element which operates under a simple rule set. Though attribution of junction rule formulation varies by domain, and in some cases is not attributed or formally named at all, the electrical domain attribution will suffice for the general description. That is, in an electrical circuit, Kirchoff's current law (or more generally, Kirchoff's Junction Rule) states that the current (flow) about a junction is summatively zero. This fundamental rule, in combination with the complementary effort rule, described in the edges section to follow, motivates the composition/derivation of linear systems of equations which are sufficient for analysis across all linearly constrained systems. Most simply, a junction is a multiple-input, multiple-output element derivative of a flow system. The junction will privately implement both flow and effort mutators inherited from the parent element type. Because junctions are singularly responsible for the exponentially of system complexity, they are also the primary means of flow analysis in some systems, such as that of an electrical power system. Conventionally, in such a system complex power flow is solved with reference to busses (junctions) and requires composition of an admittance matrix which is derived from the inertial, compliant, and dissipative characteristics of the power lines (edges) or per-unitized transformers which directly connect two busses.

5.3.6. Transformer and Gyrator. The transformer is an element derivative which accepts a single input (primary connection) and single output (secondary connection). Although in practice a single transformer may support multiple secondary connections, modeling which captures such complexity as the composition of simple transformers and junctions is enabled. In principle, a transformer routes power by amplifying one power conjugate and suppressing the other such that power is conserved and such that the power conjugate dependency relation is preserved across the transformation boundary. This relation holds across energy domains; thus, a multi-domain transformer can exist under other frameworks as multi-domain dynamical system element. However, for the purposes of this unified model, singular systems are bounded by the energy/resource domain in which they operate, meaning that cross-domain power conversion is not modeled by a transformer element, but rather by a user defined behavior executed in a run-time script interface which follows same principled logic. In such form, the primary side of the transformer is represented as a sink in a particular domain while the secondary side, functioning in a new energy/resource domain and system, is represented as a source. The component sides are then linked in script to bridge the domain boundary in accordance with the user specification. Under the FLASH framework, the gyrator may be modeled similarly. Gyration within domain boundaries is representable by the single-input, single-output gyrator element, while cross-domain gyration is captured via the combination of a primary domain sink and secondary domain source and scripted behavior defines the inversion of the power-conjugate dependency relation across the gyration.

5.3.7. Edges. Captured generally in Figure 5.18, the edge is a base with respect to the FLASH framework. An edge provides the means of information transfer and linkage between two connected elements. An edge cannot exist outside of the scope of a system and cannot exist without specifying exactly two connected elements. If for any reason an element ceases to exist at simulation run-time, any and all edges which directly link to that element must also cease to exist. In this way, any edge is jointly owned by the linking elements, or alternatively is singularly owned by the system containing all three elements.

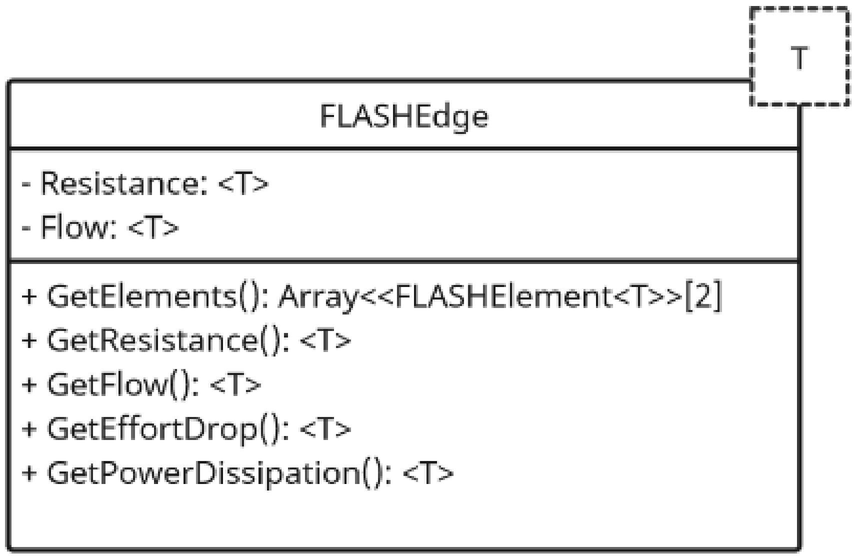


Figure 5.18. Edge Class UML

An edge explicitly links and carries the effort and flow information between connected elements and thus implicitly carries the power information. Thus, the edge is a non-abstract type which must declare and define interfaces to access element links. The edge must provide interfaces which facilitate calculation of effort and flow information across the edge. The effort interface describes the effort differential between linked elements while the flow interface describes the character of flow through the edge itself. A given edge may not be referenced directly by individual agents, but may be accessible indirectly by the agent's nodal/elemental reference.

6. INDUSTRY SOLUTIONS

6.1. DOMAIN-CENTRIC SOLUTIONS

6.1.1. PowerWorld Simulator. PowerWorld is an interactive electrical power system simulation software suite designed to simulate high voltage power system operation and control over a variable and end-user mutable time range spanning from seconds to days, claiming high effectiveness in power flow analysis with the capability to efficiently solve electrical system power flow models with up to 250,000 busses [55].

A scenario (termed *case*), is defined by the user through graphical representations of circuit elements, including active, passive, and protective, which are linked by ports into busses in representative series and/or parallel combinations. Selection of a particular element allows manipulation of on-screen character state representations as well as data mutation for calculation of nominal flow conditions at simulation initialization. The two-dimensional graphical case context lends itself to reasonable scalability representations, only becoming cumbersome, but not necessarily prohibitive, in cases containing high numbers of busses.

At simulation run-time, the graphical scenario representation of the system includes animation, the variable speed movement (per current) of dynamically sized (per voltage) green arrows across graphical edges; thus, depicting the flow of power from electrical generators to electrical loads about the system. By selecting particular elements at run-time, system state is mutable by the end-user, allowing protective breakers to be opened and generation sources to increase or decrease supply. Further, the calculated admittance matrix may be updated directly, such that a user may apply multiple faults simultaneously, a useful analysis technique for determination of voltage collapses [56]. Further a multi-command script interface supports the run-time execution of script commands or execution of auxiliary files containing scripts at simulation, allowing the user to repeatably and programmatically introduce system changes such as faults and loads. No post-run-time record of sequential state is described, limiting guarantees on run repeatability for a particular OITL case.

6.1.2. PipeFlow. Stated as being used by organizations in over one-hundred countries [57], PipeFlow Expert is an industry leading software application designed for the purpose of modeling, designing, analyzing, and solving nominal rate of flow and headloss about pipe systems. The tool includes support for open and closed loop system representations, multi-pump cases (series and parallel) and control elements such as valves. Other elements include junctions such as tees, bends, and other fittings, as well as supply sources and discharge points. PipeFlow is designed to models systems with large numbers of flow and discharge points and includes support to account for pumping requirements and variable elevation. Built in support is included for a variety of fluid models.

Like PowerWorld, PipeFlow supports scenario definition in a graphical scenario space. However, due to the relative spatial importance the scenario space supports pseudo-three-dimensional (isometric) representations of systems to reflect characteristics such as pipe length, diameter, and elevation — each being a fundamental character on overall fluid system behavior. Selection of a system element in the scenario space includes a description of the element properties, allowing for the user to mutate certain operational properties to affect nominal flow calculation. Edges are themselves pipes and include a full set of characteristics.

Systems are solved via internal software calculation per industry standard methods [57], described previously, allowing end-users to view the nominal conditions of operation under the specified constraints including calculated values of flow, velocity, and entry/exit pressure as well as flow type and Reynolds numbers. Compared to electrical power transmission systems, typical fluid systems are less prone to drastic acute changes (faults) due to robust design and construction as well as leakage tolerance for many chemically and ecologically stable/inert fluids, thus Operator-In-The-Loop (OITL) interaction in such a fluid system simulation — in isolation — is not of particular relevance for this domain, as changes in system character, such as opening and closing a particular valve, leads to a new nominal system state with relatively low concern for transience. Therefore, PipeFlow adequately meets the combined need for dynamic system control and system post-time data analysis by the same function for the purposes of fluid flow, and by similar reasoning does not require a script interface for run-time dynamical state change.

While PipeFlow meets the needs of industry with respect to fluid system assembly, mutation, and analysis, the implementation does not lend itself to other dynamical energy domains such as electrical or thermal system flow analysis in which fault consequences are realized acutely and in many cases amount to significant danger to human life as well as financial implications and, particularly in the case of electrical transmission, cascading failure as a result of overwhelming transience. This domain solution format remains compatible with the proposed extension in that calculations and terminology remain unchanged, to facilitate translation for the extensions part in compatibility, while the underlying framework requirement — agent based, time and/or event interval simulation run-time intervals — facilitates state dynamic modeling over time by design. Thus, identical state calculation in the fluid domain when implemented in an AFSM with the proposed FLASH framework must execute identically to that of PipeFlow by definition.

6.2. GENERALIZED MODELING AND SIMULATION SOLUTIONS

6.2.1. Anylogic. Anylogic at its core is a versatile tool in generalized simulation and modeling which includes proprietary and community extensions to facilitate design and analysis of spatially accurate and constrained models by way of 2D and/or 3D scenario definition. Anylogic is a leading simulation modeling software, utilized worldwide by over 40 percent of Fortune 100 companies [58]. In this way, models of the physical kinetic state of matter (resources, batches) as well as that of continuous flow system initial states may be visualized intuitively with direct parity to real-world counterparts. This intuitive visualization lends itself to adoption in particular markets as reflected by the measure of industry penetration by Anylogic into fields such as supply chain management, transportation and logistics, manufacturing, mining, and warehouse operations [58]. Additionally, partial native support is provided for modeling and simulation of fluid resource systems [59].

Anylogic simulation run-time conditions are dynamic — supporting Monte Carlo, virtual, and real-time (wall-clock) simulation an analysis of system models, agent-based models, or simultaneous combinations of both for a particular scenario with spatial visualization, run time script execution support, and OITL direct interface control. This combination of utility lends itself to powerful capability and verifiable high-fidelity models as a result of spatial constraints and observable material kinetics. Anylogic further supports powerful behavioral scripting through direct functionally scoped blocks which directly accept object-oriented Java from the end-user.

This level of control and utility, while powerful, is in many ways unapproachable in comparison to domain-centric flow system analysis tools which for a particular domain suffice and are recognized as industry standards. To bridge that divide, Anylogic offers proprietary software extension packages to accompany the baseline software suite which incorporate pre-built utilities for convenient spatial modeling of dynamical systems such as road traffic networks, port, and terminal operations, building and street foot-traffic congestion and more. These optional pay-to-access feature sets include related model implementations which allow users to see the features in action quickly. Given the exponential nature in complexity inherent to interdependent system modeling, this is level of work is required for mass adoption and is notable as a leading strategy for multi-industry application and appeal. The nature of logistical and materially discrete kinetic systems which lend themselves optimally to the visualization utilities provided by Anylogic are however much less relevant in the case of continuous flow systems.

Some applicability holds for fluid energy flow systems models, while such utilities are much less relevant — and in some sense cumbersome/inflated — for the purposes of energy flow modeling for energy domains in which the transfer itself is not observable to the human eye. Such is the case with most real-world, large scale electrical and thermal energy transfer systems. This is reflected in the relative lack of adoption in various fluid flow industries and correlates with the relative incompleteness of functionality native to Anylogic with respect to modeling such systems. This cyclical pattern of direct interdependency and circularity holds for electrical systems modeling and industry adoption — Anylogic does not provide native support for electrical systems modeling, nor is there significant adoption of Anylogic into the industry. A notable deficit in the Anylogic base feature set is a native run-time event sequence capture, meaning that observable behaviors for a given OITL simulation run may not be fully available for post-time analysis, a relative point of need for completion of the design-test-analyze-improve M&S life cycle. Reason holds; however, that an extensible and script compatible framework such as Anylogic remains, in principle, a candidate AFSM for implementation of the FLASH framework based on meeting most all AFSM requirements per Chapter 5.1.

6.2.2. Matlab and Simulink. Matlab, much like the baseline Anylogic core framework, is a mathematical computational engine based on a proprietary script language bearing syntactic similarities to other more general purpose script languages such as Python, but with the additional provision of a fully integrated MathWorks proprietary development environment — the Matlab software application suite [60]. Simulink, a multi-domain modeling and simulation environment, also designed, released, and maintained by MathWorks, extends native Matlab functionality primarily through support of recursive block system architecture and port linkage as a form of programmatic input to the Matlab engine [61].

By its nature as a script-based language and utility, Matlab lends itself to this generalized flow system framework extension. Simulink further introduces native visual and spatial scenario design to facilitate scalable models through behavioral containerization of scripts written in the Matlab script language — a syntax familiar to a vast number of professionals across scientific and engineering industries. Required OITL support is then native, but complex domain system model translation from domain tools into Matlab remains non-trivial given baseline capabilities. As with Anylogic, Matlab is limited in comparison to the extensive native analysis capability provided by domain-centric tools such as PowerWorld when compared as a means of composing and analyzing electrical power systems. Again, a fundamental justification for the adoption of a unified abstract flow system framework is thus apparent.

6.3. OBSERVED PATTERNS

Software adoption and implementation of a common language paradigm is not the natural tendency of industry [62]. Rather, as evidenced by the mass adoption of AGILE software development methodologies, the nature of intelligent life is to solve problems specifically as they present themselves, then quickly move on having alleviated apparent pains [63, 64]. In the context of complex problem spaces best suited to algorithmic resolution, this tendency lends itself to organic software implementations [65]. As with many organic phenomena, when studied after iterative and collaborative composition — the analog of long term growth for software — patterns emerge [66]. This emergence gives rise to software refactoring — the decomposition and reassembly of software into an evolved form — to optimize, simplify, and unify shared character.

Reiterating again through the lens of existing industry solutions, organic patterns now emerge, presenting themselves in the realm of complex flow domain systems architectures and solution spaces such that meaningful interdependency modeling is prohibitive. Instead, assumptions are made to suggest that requisite resource and energy dependency demand is always met for generation of energy and resources in other domains. In other words, meaningful representations of energy transformations and gyrations degenerate to assumption — ironically as a direct result of dependencies and constraints on both time (manpower) and money.

Consider again the tools described previously. PowerWorld perceived to be an optimal tool for electrical power systems analysis. This is so because PowerWorld is expressly designed to support modeling, design, and control of complex electrical power transmission and distribution networks [67]. The software serves that function well enough to reasonably claim a title as a standard of industry, but never the less makes assumptions about the means of electrical power generation rather expecting power engineers to describe the energy transformations and gyrations by which such electrical power is generated. This is understandable as complete analysis of the power system itself does not require those dependency descriptions. This exact pattern holds for other energy flow domains and their solutions — PipeFlow for fluid systems analyses — and permits depth focused progress about the particular domain [68]. within monetary and temporal constraints. Meanwhile as the field of systems engineering grows from necessity (particularly systems-of-systems engineering) systems engineers and academics alike recognize the common patterns, and propose abstractions which are sound in principle, but unapproachable in industry. Thus, abstract modeling frameworks are not dominant in industries of singular energy focus because adequate computational tools and solutions already exist and are prolifically adopted throughout the particular problem space.

Thus, systems engineers implement abstract frameworks in software and even, in a few cases, bridge the abstraction layer for particular solution spaces, such as that of Anylogic in its adoption throughout observable material kinetics and dynamics as an industry tool of choice [58], including designing native capabilities to support fluid flow system dynamics. However, through direct use of such capability in comparison to analogous capabilities in PipeFlow, Anylogic is vastly inferior to PipeFlow in that respect. A particular limitation of note in Anylogic pipe networks is a limited, element-centric, control scheme wherein flow is not determined by holistic fluid system analysis because the data required for such analysis is not required as input for pipe network construction and no governing system is necessarily defined. For example, pipes do not require a diameter or coefficient of friction to operate. Rather, sources define flow rates and flow control is then determined by simplistic methods such as explicitly definable percentages through particular outlets at junction points, or the default behavior of routing all flow into the first connected pipe until capacity is reached, at which point flow spills over into additional connections, repeating the overflow behavior as each connected pipe reaches flow capacity if such a limit is provided [69]. This attempt to bridge the gap may only be sufficient for the purposes of a multi-domain modeler using Anylogic where only a degenerative/basic fluid system behavior is required in comparison to the real-world fluid network the model attempts to describe [69]. In this case, a dependency is modeled, but not an interdependency — a key difference. For an interdependency to be modeled meaningfully, not only must the logistical network model depend on the state of the fluid flow model, but also the fluid flow model must depend on the state of the logistical model at relevant/applicable points of interface. The shortcoming is clear and motivates the lack of mass adoption of tools such as Anylogic in fluid flow system analysis spaces. Simply, the interface by which the fluid engineer defines systems in high-fidelity using PipeFlow is not easily translatable into Anylogic, inhibiting adoption.

The FLASH paradigm described is effective as a platform for industry adoption only if derived types inheriting from the abstract interface adhere to two primary requirements. First, the model definition form/scenario definition must have near parity in the way in which the user defines the system when using the domain specific tools of the particular industry. Second, the computational technique/method by which the system flow is solved must explicitly, unambiguously, overtly be that which is recognized as an industry standard within that particular domain (i.e., Hardy Cross for fluid system flow analysis, Newton-Raphson or Gauss-Sidel for electrical systems, etc.). Only then are derived types inheriting from the abstract framework extension sufficiently approachable to highly domain focused subject matter experts. Assuming that barrier is overcome, system-centric model implementations are then highly portable as they employ the language, structure, and methods of the domain from which they are motivated. Only through deliberate framework evolution in this form is meaningful, scalable interdependency modeling accessible by means of agent-based script behaviors.

7. CONCLUSIONS

7.1. CAPTURE

This work body highlights a mounting challenge in simulation and modeling — the ever increasing complexity and interdependency of critical infrastructure systems which support and define modern society are represented in modeling and simulation by flow systems models in combination with a domain knowledge barrier which makes current unified energy system modeling (FUMES) techniques unapproachable in industry in the face of practical (often fiscal) operational constraints. Mono-domain models — the de facto industry standard — are supported by domain particular methods/techniques which generally pursue simplistic (often linear) solutions through real-time gathered data and analyses or through simplified node and loop character analyses based on potential (effort) and spatial resource transfer over time (flow). Such methods constitute the simplest form of flow system model frameworks. Therefore, mono-domain frameworks are heavily adopted by industry, leading to many successful domain-specific proprietary software solutions. Industry experts are then trained, in a terminological sense, in system M&S at the level of implementation, making transition into multi-domain frameworks for higher complexity inter-dependent system analyses duly out of reach in the sense that currently accepted and industry standard software products such as those considered first in Chapter 6 do not provide the means to model such interdependencies and even if they did the standardized terminological abstraction used in academically reputable multi-energy-domain analysis frameworks (FUMES) would be the assumed interface language of system model composition. Further, such domain-spanning solutions (FUMES) require element-wise decomposition and re-assembly making translation for non-trivial systems into new software environments/scenario formats similarly prohibitive based on the risk of human error inherent to and time commitment required by such translation. A similar case is made to highlight barriers which impede industry adoption of multi-domain dynamical system analysis by means of more abstract M&S tool suites such as those considered later in Chapter 6. These abstract M&S frameworks; however, are examples of the baseline frameworks (AFSMs) with-which the proposed FLASH framework extension may be implemented.

To overcome this multifaceted challenge, a new hybrid framework extension — FLASH — is proposed in Chapter 5. The FLASH framework leverages advantages of system architecture — primarily portability without requiring translation and natural data availability and management in simulation — in combination with independent and containerized operational behavior inherent via agent-based architecture. These otherwise disjoint architectures are unified through a nodal/elemental reference pattern which grants state ownership (but not instance ownership) of zero or more individual system elements, each of which may belong to one of many systems within a singular, particular energy or resource domain. This mono-domain characterization of individual systems is optimized in implementation and compatibility based on abstract interface requirements described in the underlying framework description. Such interfaces allow implementers (i.e. software engineers in collaboration with domain experts) to translate requirements into particular domain(s); subsequently allowing users of said implementation to define domain systems models (in scenario) terminologically in identical or near identical form to that of current domain-specific industry solutions. Thus, no matter the medium by which a particular system type is defined under the framework extension, any derivative system model may be ported near trivially across any AFSM which implements the same FLASH architecture while existing models (particularly highly complex, interconnected, and trusted models of real-world critical infrastructures) may be migrated 1-to-1 (without translation) into a simulation space which includes a complete implementation of the FLASH framework in the relevant, fully-supported domain.

With a palatable method in reach by which a complex mono-domain flow system model may be ported across scenario spaces (i.e., across disjoint software suites), the extension then facilitates cross-domain system interdependency modeling with flexibility, compartmentalization, and user-defined fidelity, through use of a run-time executable script interface. Such an interface is to be defined and supported by the underlying AFSM but extended with direct data access and mutation script interfaces into abstract and derivative system type definitions in software. Through this propagation of data accessibility into the scenario space a singular agent, which defines and owns system element referential components may execute scenario defined behaviors in a simulation run-time script context. Such behaviors define logic and flow control functionality as provided by the end user which, having access and mutation rights to component referenced system element data, define the character of energy and resource transformations and gyrations between and among energy domain boundaries. Resultant state changes in a particular system element signal the simulated system instance to recalculate power and flow state at a system-defined interval. This hybrid delegation of responsibility between multi-domain agents and mono-domain systems for state and dependency resolution at simulation run-time lends itself to portability, scalability, and adoption by compartmentalizing interdependency interactions in end-user defined scripts and allowing end users to define and analyze particular systems using industry standardized terminology and flow analyses techniques.

7.2. IMPACT

This novel framework fundamentally differs from other proposed solutions. A successful implementation of the FLASH framework constrains systems models within rigid, familiar domain boundaries, only enabling dependency coupling at run-time. A primary implication of the boundary is realized in the necessarily simplified description of the transformer and gyrator (5.3.6). Uniquely simple and intuitive in comparison to FUMES counterparts, these element representations are significantly more approachable from an arbitrary domain entry point. The FLASH framework's discrete boundary scopes each system such that the conventional (often linear) means of mono-domain system analyses are directly accessible in comparison to more complex multi-domain analyses — which often require iterative calculation of partial differential equations.

As stated previously, a rigid domain boundary guarantees that any given system-type is portable to and from it's origin domain simulation and modeling space (i.e. between current industry tools and AFSMs implementing FLASH). Enforcement of the domain boundary allows transportation of a system discretely into a new simulation space with immediate validity. Only at simulation run-time do power and energy interdependencies manifest through script. By this model decoupling, initial steady-state may be achieved even where interdependencies are mapped when nominal defaults are provided at boundary points (i.e. source/sink and transformation/gyration points) — a direct reflection of nominal operation in a mono-domain simulation space. These patterns are fundamental to the viability of the framework.

FLASH architecture is designed with the acknowledgment that interested and qualified end users may only be assumed to be familiar with the analysis techniques that govern their primary domain of focus. Therefore, the assumption is made that users will rarely claim comparable familiarity with system structures and behaviors in multiple required domains. To overcome this, portability via domain-boundary enforcement in the scenario space paired with inter-dependency coupling in the simulation space yields an approachable, intuitive, and — critically — familiar pattern. FLASH adoption may revolutionize and catalyze large scale simulation and modeling of resource, energy, and power systems by building on existing software solutions. There-by, measurable progress into this deeply complex and vast problem space may yield insights, critical to the national interest, into the constitution and survivability of existing and proposed critical infrastructures.

APPENDIX

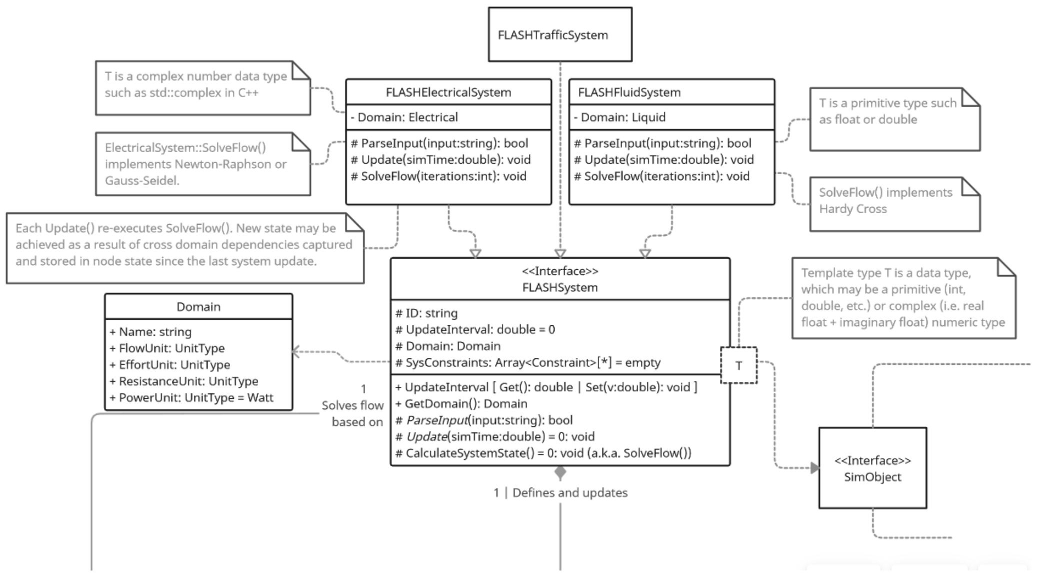
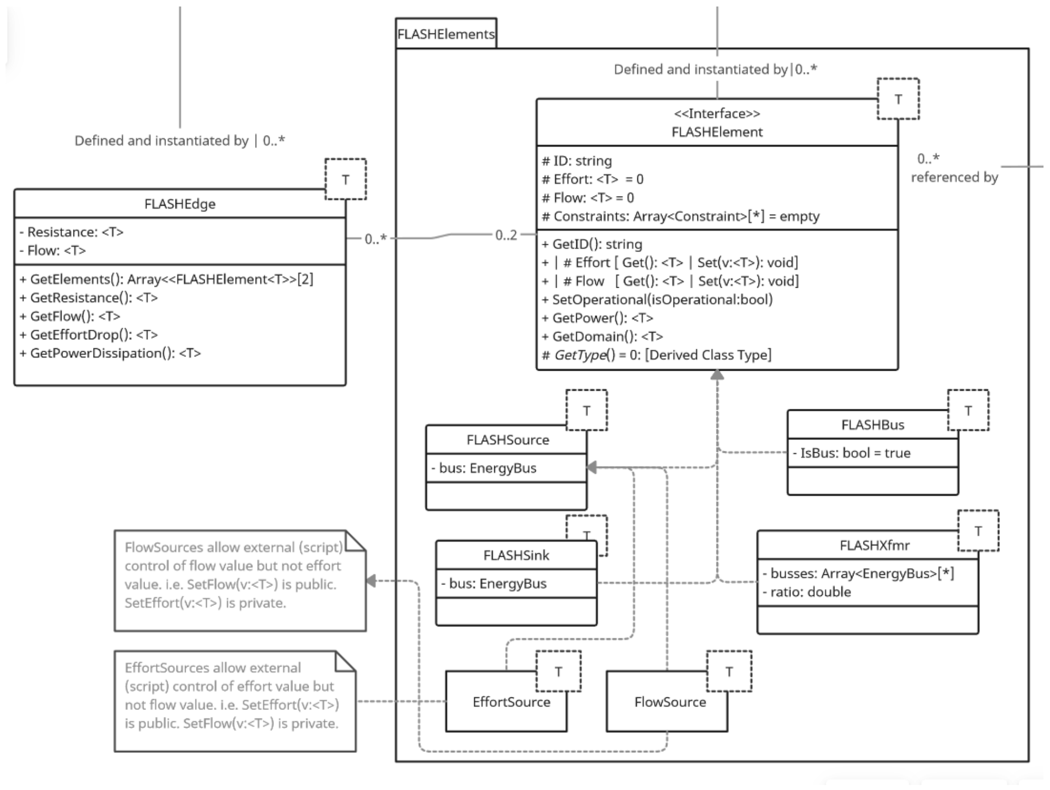

FLASHTrafficSystem
FLASHElectricalSystem
- Domain: Electrical
# ParseInput(input:string): bool
# Update(simTime:double): void
# SolveFlow(iterations:int): void
FLASHFluidSystem
- Domain: Liquid
# ParseInput(input:string): bool
# Update(simTime:double): void
# SolveFlow(iterations:int): void
T is a complex number data type such as std::complex in C++
ElectricalSystem::SolveFlow() implements Newton-Raphson or Gauss-Seidel.
Each Update() re-executes SolveFlow(). New state may be achieved as a result of cross domain dependencies captured and stored in node state since the last system update.
T is a primitive type such as float or double
SolveFlow() implements Hardy Cross
<<Interface>>
FLASHSystem
# ID: string
# UpdateInterval: double = 0
# Domain: Domain
# SysConstraints: Array<Constraint>[*] = empty
+ UpdateInterval [Get(): double | Set(v:double): void]
+ GetDomain(): Domain
# ParseInput(input:string): bool
# Update(simTime:double) = 0: void
# CalculateSystemState() = 0: void (a.k.a. SolveFlow())
Template type T is a data type, which may be a primitive (int, double, etc.) or complex (i.e. real float + imaginary float) numeric type
T
Domain
+ Name: string
+ FlowUnit: UnitType
+ EffortUnit: UnitType
+ ResistanceUnit: UnitType
+ PowerUnit: UnitType = Watt
1
Solves flow based on
1 | Defines and updates
<<Interface>>
SimObject

Figure A.1. FLASH System UML Complete



FlowSources allow external (script) control of flow value but not effort value. i.e. SetFlow(v:<T>) is public. SetEffort(v:<T>) is private.

EffortSources allow external (script) control of effort value but not flow value. i.e. SetEffort(v:<T>) is public. SetFlow(v:<T>) is private.

Figure A.2. FLASH Element UML Complete

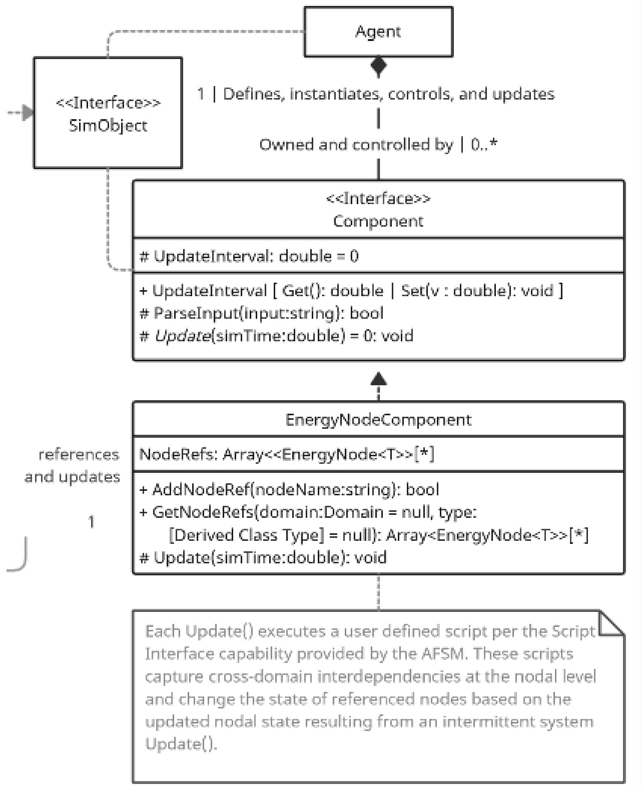

Agent
<<Interface>>
SimObject
1 | Defines, instantiates, controls, and updates
Owned and controlled by | 0..*
<<Interface>>
Component
# UpdateInterval: double = 0
+ UpdateInterval [Get(): double | Set(v : double): void]
# ParseInput(input:string): bool
# Update(simTime:double) = 0: void
EnergyNodeComponent
NodeRefs: Array<<EnergyNode<T>>[*]
+ AddNodeRef(nodeName:string): bool
+ GetNodeRefs(domain:Domain = null, type: [Derived Class Type] = null): Array<EnergyNode<T>>[*]
# Update(simTime:double): void
references and updates
1
Each Update() executes a user defined script per the Script Interface capability provided by the AFSM. These scripts capture cross-domain interdependencies at the nodal level and change the state of referenced nodes based on the updated nodal state resulting from an intermittent system Update().

Figure A.3. FLASH Agent UML Complete

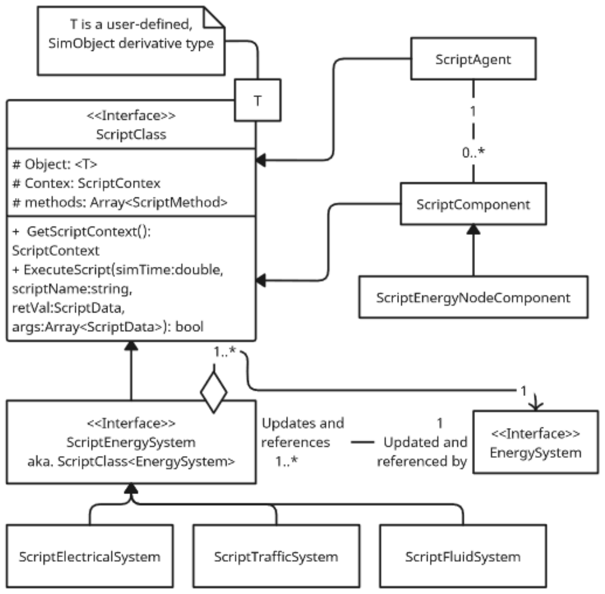

T is a user-defined, SimObject derivative type
T
<<Interface>>
ScriptClass
# Object: <T>
# Contex: ScriptContex
# methods: Array<ScriptMethod>
+ GetScriptContext(): ScriptContext
+ ExecuteScript(simTime:double, scriptName:string, retVal:ScriptData, args:Array<ScriptData>): bool
ScriptAgent
1
0..*
ScriptComponent
ScriptEnergyNodeComponent
1..*
1
<<Interface>>
ScriptEnergySystem
aka. ScriptClass<EnergySystem>
Updates and references
1..*
1
Updated and referenced by
<<Interface>>
EnergySystem
ScriptElectricalSystem
ScriptTrafficSystem
ScriptFluidSystem

Figure A.4. FLASH Script UML Complete

REFERENCES

[1] Lin Zikui and Guo Xuan. The regulation model of pharmaceutical logistics based on the supervision of flow. In *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)*, volume 2, pages 1055–1059, 2010. doi: 10.1109/ICLSIM.2010.5461117.

[2] A. D. Bain. Surveys in Applied Economics: Flow of Funds Analysis. *The Economic Journal*, 83(332):1055–1093, 12 1973. ISSN 0013-0133. doi: 10.2307/2230842. URL `https://doi.org/10.2307/2230842`.

[3] Christian Federau, Max Wintermark, Soren Christensen, Michael Mlynash, David G. Marcellus, Guangming Zhu, Blake W. Martin, Maarten G. Lansberg, Gregory W. Albers, and Jeremy J. Heit. Collateral blood flow measurement with intravoxel incoherent motion perfusion imaging in hyperacute brain stroke. *Neurology*, 92(21): e2462–e2471, 2019. ISSN 0028-3878. doi: 10.1212/WNL.0000000000007538. URL `https://n.neurology.org/content/92/21/e2462`.

[4] Jacobs, Riannea, Teunis, and Kassteele. Tracing the origin of food-borne disease outbreaks: A network model approach. *Epidemiology*, 31(3):327–333, 2020. doi: 10.1097/EDE.0000000000001169.

[5] Carmel Farage, Daniel Edler, Anna Eklöf, Martin Rosvall, and Shai Pilosof. Identifying flow modules in ecological networks using infomap. *Methods in Ecology and Evolution*, 12(5):778–786, 2021. doi: https://doi.org/10.1111/2041-210X.13569.

[6] T. Jacobs, A. Gomide, M. Kaspereit, K.-P. Zeyer, A. Kienle, and P. Hauptmann. In-line analysis of chemical reactions in micro reactors using thermal mass flow sensors. In *EUROCON 2007 - The International Conference on "Computer as a Tool"*, pages 571–574, 2007. doi: 10.1109/EURCON.2007.4400302.

[7] IEEE. Ieee recommended practice for the adoption of computer-aided software engineering (case) tools. *IEEE Std 1348-1995*, pages 1–44, 1996. doi: 10.1109/IEEESTD.1996.80818.

[8] W. Mielczarski and G. Michalik. Voltage regulation and energy consumption in electricity markets. In *IEEE Power Engineering Society. 1999 Winter Meeting (Cat. No.99CH36233)*, volume 2, pages 879–883 vol.2, 1999. doi: 10.1109/PESW.1999.747282.

[9] Abbas Ehsanfar and Babak Heydari. An incentive-compatible scheme for electricity cooperatives: An axiomatic approach. *IEEE Transactions on Smart Grid*, 9(2):1416–1424, 2018. doi: 10.1109/TSG.2016.2591507.

[10] Reynerio E. Vasquez. Selection of power transmission systems based on power losses analysis and reliability analysis for interconnection of electrical grids. In *2015 IEEE Thirty Fifth Central American and Panama Convention (CONCAPAN XXXV)*, pages 1–5, 2015. doi: 10.1109/CONCAPAN.2015.7428484.

[11] R. Wolfson. ”electricity” in energy, environment, and climate, 2nd ed. In *”Electricity” in Energy, Environment, and Climate, 2nd ed.*, number 11 in 1, 2012.

[12] Sérgio F. Santos, Matthew Gough, Desta Z. Fitiwi, André F. P. Silva, Miadreza Shafie-Khah, and João P. S. Catalão. Influence of battery energy storage systems on transmission grid operation with a significant share of variable renewable energy sources. *IEEE Systems Journal*, pages 1–12, 2021. doi: 10.1109/JSYST.2021.3055118.

[13] Ahmed Yousuf Saber and Ganesh Kumar Venayagamoorthy. Efficient utilization of renewable energy sources by gridable vehicles in cyber-physical energy systems. *IEEE Systems Journal*, 4(3):285–294, 2010. doi: 10.1109/JSYST.2010.2059212.

[14] J. Duncan Glover, Mulukutla S. Sarma, and Thomas Overbye. Power system analysis and design. In *Power System Analysis and Design*, volume 6, 2016. ISBN 1305632133.

[15] Wanjun Zhang, Feng Zhang, Jingxuan Zhang, Jingyi Zhang, and Jingyan Zhang. Study on system recognition method for newton-raphson iterations. In *2018 International Computers, Signals and Systems Conference (ICOMSSC)*, pages 737–742, 2018. doi: 10.1109/ICOMSSC45026.2018.8941745.

[16] Shihu Shu, Suiqing Liu, Xuefeng Wang, Liang Yu, Shihu Shu, Dong Zhang, and Mingqun Meng. Determination and applications of water age in distribution system. In *2010 International Conference on Mechanic Automation and Control Engineering*, pages 1918–1921, 2010. doi: 10.1109/MACE.2010.5536510.

[17] Jijun Gao, Huaidong Zhou, Gaofeng Zhao, and Xiaoru Liu. The concentration and distribution of copper(cu), arsenic(as), cadmium(cd) and mercury(hg) in drinking water of beijing city. In *2011 International Symposium on Water Resource and Environmental Protection*, volume 2, pages 1241–1243, 2011. doi: 10.1109/ISWREP.2011.5893241.

[18] Xiaohong Yan, Song Li, and Chunchun Hu. Design and implementation of the changjiang water resource commission's water intake remote monitoring and controlling system. In *2012 International Symposium on Geomatics for Integrated Water Resource Management*, pages 1–5, 2012. doi: 10.1109/GIWRM.2012.6349540.

[19] Shihu Shu, Suiqing Liu, and Dong Zhang. Investigation and control strategies of lead concentration in drinking water distribution systems. In *2010 International Conference on Mechanic Automation and Control Engineering*, pages 4151–4154, 2010. doi: 10.1109/MACE.2010.5535660.

[20] Zhai Chengwu, Huang Qiang, Chang Jianxia, and Gao Fan. The study of water resources reasonable allocation of baoji area in wei river with considering the ecology base flow. In *2011 International Symposium on Water Resource and Environmental Protection*, volume 1, pages 816–818, 2011. doi: 10.1109/ISWREP.2011.5893132.

[21] D. Trinchero, R. Stefanelli, A. Galardini, and B. Fiorelli. Wireless sensors for a wire-independent analysis of fluid networks. In *2009 IEEE Sensors Applications Symposium*, pages 105–108, 2009. doi: 10.1109/SAS.2009.4801787.

[22] Sherin A Kochummen, N E Jaffar, and Nasar A. Model reference adaptive controller designs of steam turbine speed based on mit rule. In *2015 International Conference on Control Communication & Computing India (ICCC)*, pages 7–11, 2015. doi: 10.1109/ICCC.2015.7432861.

[23] Muhammad Umar Afzaal, Muhammad Hammad Khan, and Arshad Elahi. Water flow patterns generation model for grid connected run-of-river hydro power plant. In *2022 IEEE International Conference on Environment and Electrical Engineering and 2022 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, pages 1–6, 2022. doi: 10.1109/EEEIC/ICPSEurope54979.2022.9854662.

[24] Cynthia Juwita Ismail, Anna Reani, Novelita Wahyu Mondamina, and Vini Charloth. Rehabilitation at 2×400mw indonesia coal-fired power plant: Emission reduction and cost-effectiveness. In *2017 IEEE 7th International Conference on Power and Energy Systems (ICPES)*, pages 117–120, 2017. doi: 10.1109/ICPESYS.2017.8215932.

[25] Vivek Patel, Balaram Saha, and Kalyan Chatterjee. Fuel saving in coal-fired power plant with augmentation of solar energy. In *2014 International Conference on Power, Control and Embedded Systems (ICPCES)*, pages 1–5, 2014. doi: 10.1109/ICPCES.2014.7062811.

[26] Ion Pană, Florinel Dinu, and Gabriela Bucur. Analysis of a filter used in the natural gas gathering system: Recommendations on the measuring of the pressure drop. In *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6, 2016. doi: 10.1109/ECAI.2016.7861071.

[27] A. Cioloca and R. Both. Modelling versions and simulations of the cryogenic air distillation column. In *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, pages 1–6, 2014. doi: 10.1109/AQTR.2014.6857925.

[28] Yu Zhu, Xinggao Liu, and Zhiyong Zhou. Optimization of cryogenic air separation distillation columns. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 7702–7705, 2006. doi: 10.1109/WCICA.2006.1713466.

[29] Elsevier. 2.05 - nanofiltration operations in nonaqueous systems. In Enrico Drioli and Lidietta Giorno, editors, *Comprehensive Membrane Science and Engineering*, pages 91–113. Elsevier, Oxford, 2010. ISBN 978-0-08-093250-7. doi: https://doi.org/10.1016/B978-0-08-093250-7.00036-0.

[30] Michael Judge and Greg Daniels. King of the hill. In *King of the Hill*, number 23 in 2, 2009.

[31] Edgar C. Portante, Stephen M. Folga, Gustav Wulfkuhle, Brian A. Craig, and Leah E. Talaber. New madrid and wabash valley seismic study: Simulating the impacts on natural gas transmission pipelines and downstream markets. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 2867–2878, 2009. doi: 10.1109/WSC.2009.5429241.

[32] Xiaoqiang Zhao, Wei Li, and Yan Wan. Storage & transportation scheduling model and algorithm of petroleum products based on network. In *2007 IEEE International Conference on Automation and Logistics*, pages 2052–2055, 2007. doi: 10.1109/ICAL.2007.4338912.

[33] Xiaodao Chen, Yuchen Zhou, Hong Zhou, Chaowei Wan, Qi Zhu, Wenchao Li, and Shiyan Hu. Analysis of production data manipulation attacks in petroleum cyber-physical systems. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7, 2016. doi: 10.1145/2966986.2980091.

[34] John Manfreda. The real history of fracking: Oil, bombs and civil war. In *Editorial*, volume 1, 2019. URL *https : //oilprice.com/Energy/Crude − Oil/The − Real − History − Of − Fracking.html.*

[35] Crude oil distillation and the definition of refinery capacity, 2022. URL `https://www.eia.gov/todayinenergy/detail.php?id=6970`.

[36] Sundar and About SundarMechanical Engineer. Hardy cross method procedure for pipe network analysis, Sep 2022. URL *https : //extrudesign.com/hardy − cross − method − procedure − for − pipe − network − analysis/.*

[37] Nick Connor. What is darcy-weisbach equation - definition, Jun 2019. URL *https : //www.thermal−engineering.org/what−isdarcy−weisbach−equation−definition/.*

[38] Hazen-williams friction loss equation - calculating head loss in water pipes, 2022. URL *https : //www.engineeringtoolbox.com/hazen − williams − water −* d_7*97.html.*

[39] A. Hegyi, Bart De Schutter, and J. Hellendoorn. Optimal coordination of variable speed limits to suppress shock waves. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):102–112, 2005. doi: 10.1109/TITS.2004.842408.

[40] Alessio Tesone, Angelo Coppola, Luca Di Costanzo, Luigi Pariota, and Gennaro Nicola Bifulco. Route guidance systems based on the macroscopic fundamental diagram concept: a simulation-based case study in the city of portici. In *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, pages 1–6, 2021. doi: 10.1109/EEEIC/ICPSEurope51590.2021.9584783.

[41] Pu Wang, Jiyu Lai, Zhiren Huang, Qian Tan, and Tao Lin. Estimating traffic flow in large road networks based on multi-source traffic data. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5672–5683, 2021. doi: 10.1109/TITS.2020.2988801.

[42] Rajneesh Tanwar, Rana Majumdar, Gursewak Singh Sidhu, and Abhishek Srivastava. Removing traffic congestion at traffic lights using gps technology. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pages 575–579, 2016. doi: 10.1109/CONFLUENCE.2016.7508185.

[43] Mohammad Hajiahmadi, Jack Haddad, Bart De Schutter, and Nikolas Gerolimi-nis. Optimal hybrid perimeter and switching plans control for urban traffic networks. *IEEE Transactions on Control Systems Technology*, 23(2):464–478, 2015. doi: 10.1109/TCST.2014.2330997.

[44] Solving traffic flow problems, 2022. URL `https://origindesign.com/articles/traffic-modeling`.

[45] Hang Shen and Huiyuan Jiang. Research on road construction's impact upon traffic flow. In *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, volume 2, pages 429–432, 2011. doi: 10.1109/CCIENG.2011.6008156.

[46] V. Sreedhar. Plug-in hybrid electric vehicles with full performance. In *2006 IEEE Conference on Electric and Hybrid Vehicles*, pages 1–2, 2006. doi: 10.1109/ICEHV.2006.352291.

[47] Piping and pipelines, 2022. URL $https : //www.dla.mil/Energy/About/Library/$.

[48] Thermal engineering, Jun 2019. URL $https : //www.thermal - engineering.org/$.

[49] B31.3 - process piping, 2022. URL $https : //www.asme.org/codes - standards/find-codes-standards/b31-3-process-piping/2020/print-book$.

[50] M.P. Singh, N. Subramanian, and Rajamenakshi. Visualization of flow data based on clustering technique for identifying network anomalies. In *2009 IEEE Symposium on Industrial Electronics & Applications*, volume 2, pages 973–978, 2009. doi: 10.1109/ISIEA.2009.5356304.

[51] Abolfazl Mosaddegh, Claudio A. Cañizares, and Kankar Bhattacharya. Optimal demand response for distribution feeders with existing smart loads. *IEEE Transactions on Smart Grid*, 9(5):5291–5300, 2018. doi: 10.1109/TSG.2017.2686801.

[52] Paynter. Analysis an design of engineering systems. In *The M.I.T. Press.*, 1961. ISBN 0-262-16004-8.

[53] Peter J. Gawthrop and Geraint P. Bevan. Bond-graph modeling. *IEEE Control Systems Magazine*, 27(2):24–45, 2007. doi: 10.1109/MCS.2007.338279.

[54] Arjan van der Schaft and Dimitri Jeltsema. *Port-Hamiltonian Systems Theory: An Introductory Overview*. Now Publishers Inc, 2014.

[55] Powerworld simulator, 2022. URL $https : //www.powerworld.com/products/simulator/overview$.

[56] Y. Koyama, T. Sasaki, S. Ihara, and E.R. Pratico. Voltage collapse scenario search. In *Proceedings. International Conference on Power System Technology*, volume 1, pages 344–348 vol.1, 2002. doi: 10.1109/ICPST.2002.1053562.

[57] Pipe flow software, 2022. URL `https://www.pipeflow.com`.

[58] Simulation modeling software tools & solutions for business, 2022. URL `https://www.anylogic.com/`.

[59] Oil and gas simulation software, 2022. URL `https://www.anylogic.com/oil-and-gas/`.

[60] Matlab, 2022. URL $https : //www.mathworks.com/products/matlab.html ?s_tid = hp_products_matlab$.

[61] Simulink - simulation and model-based design, 2022. URL $https : //www.mathworks.com/products/simulink.html?s_tid = hp_products_simulink$.

[62] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Bottom-up technologies for reuse: Automated extractive adoption of software product lines. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 67–70, 2017. doi: 10.1109/ICSE-C.2017.15.

[63] Parita Jain, Arun Sharma, and Laxmi Ahuja. The impact of agile software development process on the quality of software product. In *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pages 812–815, 2018. doi: 10.1109/ICRITO.2018.8748529.

[64] How do animals solve problems?, Jun 2022. URL $https : //www.jcu.edu.au/news/releases/2021/april/how - do - animals - solve - problems$.

[65] T. Grechenig and W. Zuser. Creating organic software maturity attitudes (cosma) selected principles and activities for software maturity in small and medium software enterprises. In *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.*, pages 134–143, 2004. doi: 10.1109/QSIC.2004.1357954.

[66] Hongyu Pei Breivold, Muhammad Aufeef Chauhan, and Muhammad Ali Babar. A systematic review of studies of open source software evolution. In *2010 Asia Pacific Software Engineering Conference*, pages 356–365, 2010. doi: 10.1109/APSEC.2010.48.

[67] History, 2022. URL `https://www.powerworld.com/company/history`.

[68] Pipe flow software, 2022. URL $https : //www.pipeflow.com/pipe - flow - expert - software$.

[69] Fluid library, 2022. URL $https : //anylogic.help/library - reference - guides/fluid - library/index.html\#fluid - library$.

VITA

David Corder Hinton studied with breadth focus in computer science, computer engineering, and electrical engineering at Missouri University of Science & Technology. David was awarded a Bachelor of Science in Computer Engineering with a minor in Computer Science in May 2020. Between 2019 and 2023 he completed graduate studies at Missouri S&T while employed as a simulations software engineer with various organizations in support of the United States Department of Defense. David's M.S. studies at Missouri S&T emphasized simulation and modeling and culminated in this thesis and accompanying work body. David received a Master of Science in Computer Engineering from Missouri S&T in May 2023.