systems

MDPI

Article

Model-Based Design and Formal Verification Processes for Automated Waterway System Operations

Leonard Petnga [1,*,†] and Mark Austin [2,†]

[1] Department of Civil and Environmental Engineering, University of Maryland, College Park, 20742 MD, USA
[2] Department of Civil and Environmental Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA; austin@isr.umd.edu
* Correspondence: lpetnga@umd.edu; Tel./Fax: +1-301-314-9218
† These authors contributed equally to this work.

Academic Editor: Ockie Bosch
Received: 29 March 2016; Accepted: 27 May 2016; Published: 7 June 2016

Abstract: Waterway and canal systems are particularly cost effective in the transport of bulk and containerized goods to support global trade. Yet, despite these benefits, they are among the most under-appreciated forms of transportation engineering systems. Looking ahead, the long-term view is not rosy. Failures, delays, incidents and accidents in aging waterway systems are doing little to attract the technical and economic assistance required for modernization and sustainability. In a step toward overcoming these challenges, this paper argues that programs for waterway and canal modernization and sustainability can benefit significantly from system thinking, supported by systems engineering techniques. We propose a multi-level multi-stage methodology for the model-based design, simulation and formal verification of automated waterway system operations. At the front-end of development, semi-formal modeling techniques are employed for the representation of project goals and scenarios, requirements and high-level models of behavior and structure. To assure the accuracy of engineering predictions and the correctness of operations, formal modeling techniques are used for the performance assessment and the formal verification of the correctness of functionality. The essential features of this methodology are highlighted in a case study examination of ship and lock-system behaviors in a two-stage lock system.

Keywords: model-based systems engineering; formal verification; automation; modeling; waterways operation; canal systems

1. Introduction

Waterway systems have provided an economical means for transporting high-bulk (e.g., grain, steel, minerals, hazardous materials) and manufactured goods over long distances since the industrial revolution. Today, waterway systems serve as critical arteries in national and world trade. Within the U.S., for example, inland waterway systems carry large amounts of bulk cargo that would otherwise travel by truck or by rail, easing congestion on complementary surface transportation systems. These long-term benefits are now threatened by a host of problems: (1) Traffic demands on canal systems that have far exceeded initial expectations; (2) Increases in the prevalence and severity of delays caused by aging infrastructure; (3) Locks that are too small for modern ships and barges; and (4) Competition from alternatives, such as intermodal freight transportation [1–3]. As a result, it is becoming increasingly difficult to maximize system throughput and to minimize delays and operation costs through operations controlled solely by humans alone.

In our opinion, if waterway systems are to remain economically competitive well into the 21st Century, then modernization efforts will need to take advantage of information-age technologies.

Good solutions will improve traffic flows through modernization and replacement of aging infrastructure with systems that make increased use of automation either to handle tasks once handled by humans or to support and enhance human performance. This, in turn, points to a strong need for improved awareness and performance through sensing, fast control response and high resilience [4,5]. Indicators of this trend can be found in the modernization of traffic management systems for the Bosporus Straits (Turkey), Tsushima Straits (Korea) and the Panama Canal (Panama). In the case of the Panama Canal, lock operations now rely on an extensive sensor network coupled with lasers and radar technologies to operate and ensure fault tolerance [6,7].

This paper departs from the piecewise approach of modernizing narrow waterway systems and argues for the use of sound system thinking and systems engineering techniques for their design and automated operation. The centerpiece of our work is model-based systems engineering (MBSE) techniques, backed by methods for formal behavioral simulation and mathematical analysis of the correctness of system functionality with respect to system goals, system performance, design space exploration and formal verification. We propose a multi-level multi-stage methodology for the model-based design, simulation and formal verification of automated waterway system operations. At the front-end of development, semi-formal modeling techniques are employed for the representation of project goals and scenarios, textual requirements and high-level models of behavior and structure. To assure the accuracy of engineering predictions and the correctness of operations, formal modeling techniques are used for the performance assessment and formal verification of the correctness of functionality.

To demonstrate the essential features of the proposed methodology, the scope of this paper will focus on an examination of these concerns for ships descending and ascending through an automated two-lock system, as shown in Figure 1. The system structure consists of Locks 1 and 2 (called L1 and L2, respectively), Gates 0, 1 and 2, Pumps 1 and 2, a controller, ships and a reservoir (R), which provides water to the lock system whenever needed. System behaviors will correspond to a mixture of physical system behaviors (e.g., gate operations, fluid flows and ship movements) and cyber system behaviors (e.g., sequences of control actions). For the purposes of illustration, let us assume that a ship is positioned in Lock 2. The down-stepping process will be accomplished in a sequence of three fluid-flow operations: Step 1: Water is transferred from Lock 2 to the reservoir; Step 2: Water is transferred from the reservoir to Lock 1 for equalization of the water level with Lock 2; and finally, after the ship is towed into Lock 1, Step 3: Water is transferred from L1 back to the reservoir R, thereby allowing the ship to be lowered to sea level, then towed out. A similar three-step process is needed for a ship ascending from the sea level to the lake.

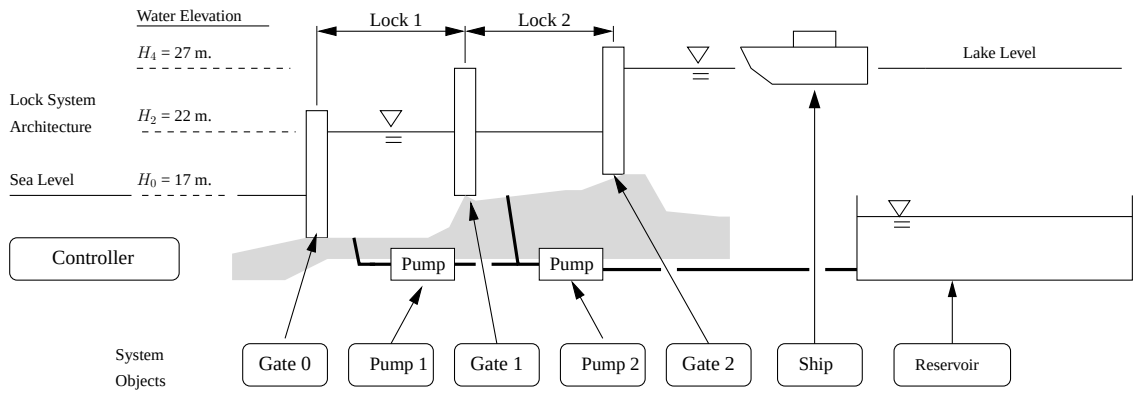

Figure 1. Simplified view of ships descending through a two-stage lock system.

The remainder of this paper is organized as follows: Section 2 describes a systems view of waterways and their operations. Section 3 describes the pathway of the development of the model-based design. The primary focus of this study is the fluid system, which is the main subsystem

of the lock operations. Section 4 covers the related theory for the engineering analysis of the coupled structure-fluid system and processes for the assessment of the correctness of system functionality with modeling checking procedures and UPPAAL [8]. Section 5 presents the essential details of the model-based design, assessment and formal validation of canal water lock operations. Finally, Section 5.5 through Section 5.7 demonstrate the use of timed automata models to formally verify the correctness of operations for the waterway system controller. MagicDraw SysML [9] is employed for the representation of the requirements and fragments of system behavior in SysML. The water lock fluid system performance will be represented as transfer equations, and the continuous system time-history simulations will be computed in OpenModelica [10]. At this point, the bridge between the continuous time-history simulation representation and the finite-dimensional real-time system (timed automata) representation is manual, with discrete block approximations of the continuous time history simulation acting as the timing constraint input for control verification in UPPAAL.

2. System View of Narrow Waterways

Whether they be man-made (canals) or natural (straits), looking at waterways from a systems perspective provides an opportunity to understand and deal with them as a whole. Our view of a system is as a collection of interconnected components having system-level functionality (as a whole) that is beyond the execution capabilities of individual components. In other words, the system is more than the sum of its parts. People, products and processes can be integral parts of a system. Thus, narrow waterways are systems that link two water arms and serve as a bridge to ships. From this perspective, they have all of the characteristics of a complex system:

A. *Boundary:* This is the physical perimeter of the waterway that separates it from its external environment. This is defined based on the specificity of the given waterway, taking into account the traffic, physical constraints, security and operation parameters. Geopolitical and environmental considerations may play important roles in delimiting the boundaries of the waterway system. We note, in particular, that most of the world's straits are shared by two or more countries, while the majority of canals are inland.

B. *Inputs and outputs:* Ships are physical entities that enter and exit the waterway through the above-mentioned boundaries. From a component and component assembly perspective, flows of ships are the inputs and outputs of our system. Whether they navigate the waterway by their own or with some help, they need and use information provided by the system controller (traffic management system) to complete the crossing.

C. *Subsystems and components:* Straits and canals have subsystems and components that are distinct. In the former, they are the "zoning" naturally created by the shape and width of the strait. In the latter, sets of lock chambers are subsystems distributed along the canal. The waterway control system regulates the traffic and/or commands the crossing operation as a core subsystem of the waterway. Depending on the analysis need, subsystems can be decomposed recursively into low-level components having well-defined repeatable functionality. We will illustrate this decomposition process in the case study section below.

D. *Connectivity:* System connectivities are defined by the physical (wired) and non-physical (wireless) connections between components. For our purposes, connectivities allow for the coordination and sequencing of actions necessary to achieve efficient and safe execution of the ship crossing process. In modernized waterway systems, connectivity can be achieved through the use of geographical information systems (GIS) and communication between the distributed set of locks in canal systems or check points in straits [11,12].

E. *Surrounding environment:* This comprises elements that are external to the system, but that might influence its functioning and/or performance. Extreme weather events, such as heavy rain, are environmental elements that can potentially affect the navigability of the waterway.

While this five-part description appears intuitive and straight forward, it lacks some critical elements that characterize a system. First, many systems have properties that only emerge when the

system is assessed as a whole. Failure to understand them can result in catastrophic consequences. In the case of waterways, understanding mechanisms through which heavy rain and pollution fill the passageway with sand and other waste is critical to ensuring they remain navigable. Second, modern canals are systems that react to sequences of incoming events, as opposed to input/output process-based systems [13]. In many cases, canals systems are safety-critical: any malfunctioning could lead to loss of life and/or property. Finally, modern canal systems are operated by embedded computers and comprise a control system that enforces the desirable behavior on the overall system.

3. Pathway of Development for Model-Based Design

The central tenet of model-based system engineering (MBSE) is the use of models as opposed to documents as the main artifacts of system development. MBSE procedures provide a formal basis for: (1) Closing the gap between what is needed and how the system will work through the development of virtual prototypes; (2) Assisting in the management of complex systems; and (3) Early and formal approaches to system validation and verification. The traditional role of engineering analysis of waterway systems has been to focus on performance, where sophisticated techniques [14–17] are justified by the adverse economics of poor system throughput. The introduction of automation into canal management expands the range of design concerns that need to be addressed. For example, a new fundamental question is: How do we know that an automated canal management system will always do the right thing? Experience [18–21] indicates that these challenges need to be addressed through a combination of mechanisms involving systematic application of decomposition, composition, abstraction and use of semi-formal and formal analysis.

3.1. Multi-Level Approach Model-Based System Design

We propose a multi-level approach model-based system design having an intricate combination of mechanisms, as shown in Figure 2.

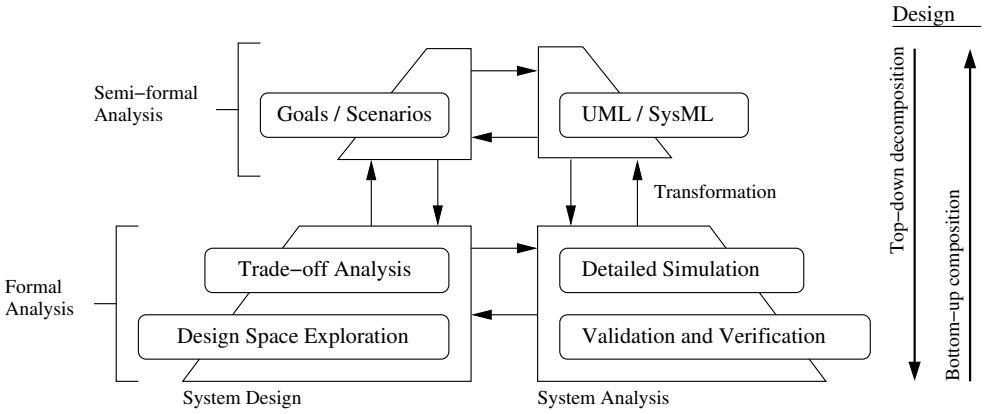

Figure 2. Multi-level approach model-based systems engineering. Semi-formal models provide a high-level view of the complete system (efficiency). Formal models provide a detailed view of the actual system (accuracy).

The pyramid structure partitions the development effort into four interrelated blocks organized into two levels. The top level contains semi-formal models capturing ideas (goal/scenarios) and preliminary designs represented in graphical languages, such as the Unified Modeling Language (UML) and the System Modeling Language (SysML) [22,23]. Together, goals and scenarios analysis and the use of UML/SysML provide the designer with a "big picture" summary of the system under development and highlight the major components, their connectivity and performance. Representations for preliminary/tentative design need to be based on semi-formal models (e.g, UML and SysML) that have a fixed syntax and semantics and, thus, can be used to communicate ideas among the

participating disciplines [22,23]. The lower level comprises models built from formal languages having precisely-defined semantics. These models provide computational support for: (1) Detailed simulation of system behavior to assess achievable levels of performance; (2) Verification of the correctness of functionality, particularly in the system control; and (3) Systematic design space exploration. Together, the combination of high- and low-level representations work to prevent serious flaws in design direction and to provide deep insight into the system behavior and functionality through formal analyses.

3.2. Pathway of Development

Figure 3 shows the pathway from an operations concept to simplified models for behavior and structure, requirements, system-level design and model checking.

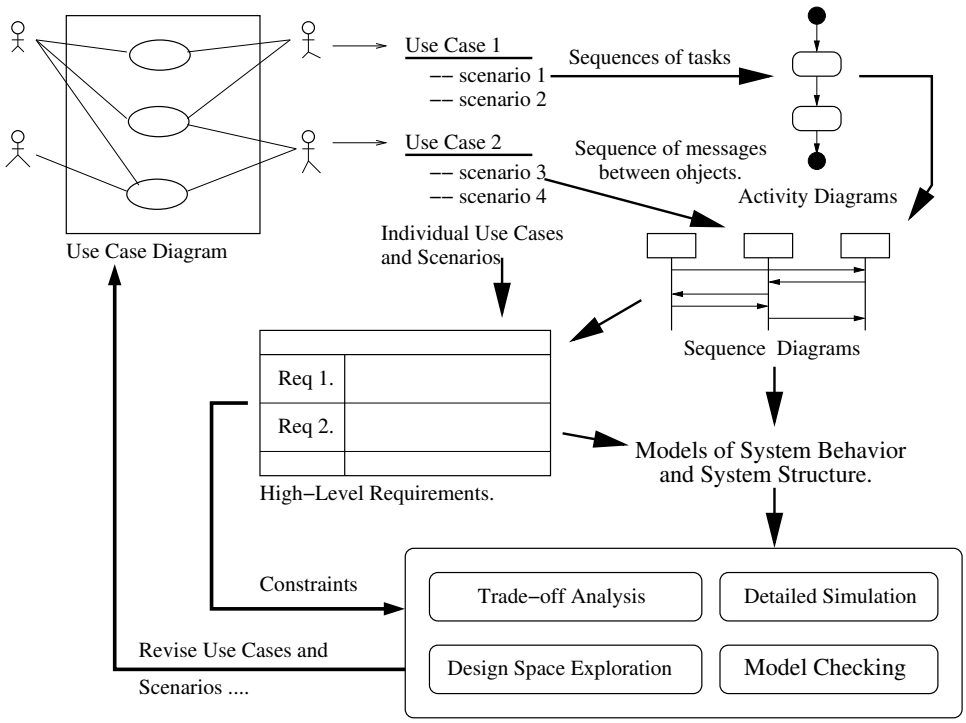

Figure 3. Pathway from operations concept to simplified models for behavior and structure, to requirements, system-level design and model checking.

The first important task is to develop a functional description for what the system will do? Since a system does not actually exist at this point, these aspects of the problem description will be written as design requirements and mathematical constraints. It is important to note that while use-cases and textual scenarios are neither requirements nor functional specifications, they imply requirements, objects and object interactions and interfaces in the stories they tell. As a case in point, when use-cases are associated with a specific class (in the system), working scenarios are, in essence, an invocation of the operations in the class. Some use-cases will correspond to only a single operation. Others will involve a set of operations, usually occurring in a well-defined sequence. Further design requirements/constraints will be obtained from the structure and communication of objects in the models for system functionality (e.g., required system interfaces). Models of behavior specify what the system will actually do; often they can be represented as networks and hierarchies of tasks, functions and processes. Models of structure specify how the system will accomplish its purpose. The system structure corresponds to collections of interconnected objects and subsystems, constrained by the environment within which the system must exist. The nature of each object/subsystem will be captured by its attributes, such as the physical structure of the design, environmental elements that will interact

with the system and the system inputs and system outputs. We create the system-level design by mapping fragments of system functionality onto specific subsystems/objects in the system structure. Thus, the behavior-to-structure mapping defines in a symbolic way the functional responsibility of each subsystem/component. In the system evaluation, the performance and characteristics of the system-level design are evaluated against the test requirements.

The heavy arrows in Figure 3 show the pathways of traceability and iterations of refinement within the model-based development. Engineers should be able to look at a requirement and understand: (1) the goals and scenarios from which the requirements emanated; and (2) the ways in which the requirement has been satisfied in the system implementation. Pathways to requirement verification can involve a multitude of analytical procedures involving (continuous system) simulation for performance assessment and formal approaches to the analysis of (discrete) control actions for the verification of the correctness of system functionality. Usually, several iterations of development will be needed to modify the system behavior, system structure, perhaps even the original operations concept, and to achieve a design that satisfies all of the system-level requirements.

3.3. Goals and Scenario Analysis

The front-end of system development is where most of the important decisions are made in terms of the project direction (and hence, the commitment of available funds) and the expectations of system functionality, behavior and cost.

Figure 4 shows the pathway from goals and scenarios into high-level requirements. The project goals come from the project stakeholders (or a customer), and may be listed very informally, as needed. A stakeholder might describe, for example, what he/she has in mind in terms of functionality, performance and cost. Since these needs may be (very) informally stated and may not have been completely thought through, several iterations of development may be required to elicit, analyze and clarify the project goals (measures of effectiveness), scenarios and requirements. The functional aspects of the project goals (*i.e.*, what the system will do) are traced to use-cases, which, in turn, can be elaborated in terms of detailed textual scenarios and SysML diagrams. The project goals that are related to performance and cross-cutting system-level considerations (e.g., reliability, security) trace directly to the textual requirements. In working from the left- to right-hand sides of Figure 4, we systematically transform an informal representation into a semi-formal representation, suitable for acting as a stepping stone to formal analysis. The final point to note is that when systems development is a team activity, requirements emanate from multiple sources (some external to the project itself). This leads to clusters of goals and scenarios, use-cases and requirements connected via one-to-many and many-to-many relationships. The latter is one of the main reasons why modeling, traceability and evaluation of cause-and-effect relationships across various stages of development is difficult.

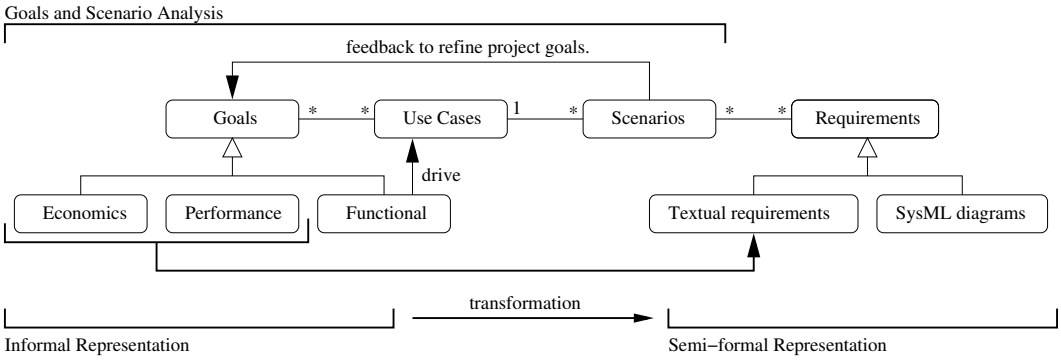

Figure 4. Transformation of goals and scenarios into high-level requirements.

4. Related Theory for Engineering System Behavior and Formal Verification Processes

This section introduces theory relating to the behavior modeling and formal validation of functionality in the canal system. Procedures for the assessment of the canal system performance will be based on variational analysis of fluid system flows and sequencing of pump and gate operations. Timed (finite state) automata and model checking procedures will be used to assess the correctness of functionality for waterway system operations.

4.1. Mathematical Modeling of the Fluid System

The desired configuration of the fluid system is the one that: (1) does not violate flow continuity constraints of fluid, *i.e.*, water in this case, and (2) maintains the fluid pressure compatibility constraints in the system. Thus, instead of the full loop variational indicator [24], we rely on the complementary form of Hamilton's principle, which results in the fluid system obeying the following set of Lagrangian's equations of motion.

$$\frac{d}{dt}\left(\frac{\partial L^*}{\partial \dot{V}_j}\right) - \frac{\partial L^*}{\partial \dot{V}_j} + \frac{\partial G}{\partial \dot{V}_j} = P_j \tag{1}$$

where G is the content of the fluid dissipators, P_j the generalized fluid pressure source of the j_{th} fluid volume coordinate; $j = 1, 2, ..., g$. The system co-Lagrangian L^* is defined based on the total co-kinetic energy of the system fluid inertances T^* and the total potential energy of the system fluid flow stores U as follows ($r \leq g$).

$$L^* = T^*(\dot{V}_1, \dot{V}_2, ..., \dot{V}_r) - U(V_1, V_2, ..., V_r) \tag{2}$$

The fluid system Lagrangian L is defined based on the total kinetic energy of the system fluid inertances T and the total co-potential energy of the system fluid flow stores U^* as follows.

$$L = U^*(\dot{\Gamma}_1, \dot{\Gamma}_2, ..., \dot{\Gamma}_m) - T(\Gamma_1, \Gamma_2, ..., \Gamma_m) \tag{3}$$

Equations (1) through Equation (3) describe the behavior of a wide range of dynamical systems. The next step is to customize the mathematical formulation to the specific details of fluid-flow operation in our canal system setup.

Step 1. The relative configuration of Lock 2 and the reservoir means that water can flow from Lock 2 to the reservoir by gravity alone. Therefore, L2 is the only volume considered; thus, $g = 1$, and T^*, U and G are defined using the reservoir capacitance C_{fR}, pipe inertance I_f and resistance R_f as follows.

$$T^* = \frac{1}{2} I_f \dot{V}_2^2 \tag{4}$$

$$U = \frac{1}{2C_{fR}} V_2^2 \tag{5}$$

$$G = \frac{1}{2} R_f \dot{V}_2^2 \tag{6}$$

We combine Equations (4) and (5) into Equation (2) and replace the result into Equation (1) along with Equation (6). We also have $P_j = P_S$, the pressure of the ship, and express the volume of the water in Lock 2 as a function of its section A_2 and water height h_2 by $V_2 = A_2 h_2$. These transformations result in the following differential equation that captures the dynamics of water flow from lock L2 to reservoir R.

$$A_2 I_f \ddot{h_2} + A_2 R_f \dot{h_2} + \frac{A_2}{C_{fR}} h_2 = P_S \tag{7}$$

Step 2. Water is transferred from the reservoir to Lock 1 for equalization of the water level with Lock 2. This step is accomplished with *nodal analysis*, which is a fluid momentum-based (Γ) approach appropriate for understanding the dynamic of the water flowing from the reservoir R to Lock L1. Lagrangian's equations of motion are written as follows:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\Gamma}_j}\right) - \frac{\partial L}{\partial \dot{\Gamma}_j} + \frac{\partial J}{\partial \dot{\Gamma}_j} = Q_j \tag{8}$$

where J is the co-content of the fluid dissipators; Q_j the generalized fluid flow source associated to the j_{th} fluid momentum coordinate; $j = 1, 2, ..., l$.

Step 3. Water is transferred between lock L1 and the reservoir. From Equation (1) through Equation (8), $l = 2$, and T, U^*, Q_j and J are defined using, in addition to the parameters defined above, the reservoir capacitance C_{fR}, section A_R, water height h_R and the water height in Lock 1 h_1 as follows.

$$U^* = \frac{1}{2}C_{f1}\dot{\Gamma}_1^2 + \frac{1}{2}C_{fR}\dot{\Gamma}_R^2 \tag{9}$$

$$T = \frac{1}{2I_f}\left(\Gamma_1 - \Gamma_R\right)^2 \tag{10}$$

$$J = \frac{2}{3R_f^{1/2}}\dot{\Gamma_R}^{3/2} \tag{11}$$

The first order linearization of Equation (11) at the connection Point B of the pipe to the reservoir (as close as possible to the bottom of the reservoir) results in the following equation.

$$J_B(\dot{\Gamma_R}) = \frac{2\dot{\Gamma_B}^{3/2}}{3R_f^{1/2}} + \left(\frac{\dot{\Gamma_B}}{R_f}\right)^{1/2}(\dot{\Gamma_R} - \dot{\Gamma_B}) \tag{12}$$

We replace Equations (9) and (10) into Equation (8) and subsequently into Equation (7) along with Equation (12). Since there is no external source of water flow to any of the two tanks, we have $Q_j = 0$. Furthermore, we have $P = \dot{\Gamma} = \frac{A}{C_f}h$. Equipped with these transformations, we compute the derivate of the resulting expression of Equation (7), respectively, to Γ_1 and Γ_R and obtain the following set of differential equations.

$$A_1\ddot{h_1} + \frac{A_1}{I_f C_{f1}}\int_0^t h_1(\tau)d\tau - \frac{A_R}{I_f C_{fR}}\int_0^t h_R(\tau)d\tau = 0 \tag{13}$$

$$A_R\ddot{h_R} + \frac{A_R}{I_f C_{fR}}\int_0^t h_R(\tau)d\tau - \frac{A_1}{I_f C_{f1}}\int_0^t h_1(\tau)d\tau + \left(\frac{\dot{\Gamma_B}}{R_f}\right)^{1/2} = 0 \tag{14}$$

With the ship now in Lock 1, a pump with pressure P_t is needed to pump the water out of L1 to the reservoir R. We perform another loop variational analysis using the same set of Lagrangian's equations of motion as Equation (1) and rewriting Equations (3) to (5). After combination, simplification and replacement of A by the section of the ship in contact with water in Lock 1, *i.e.*, A_{1S}, we obtain:

$$A_{1S}I_f\ddot{h_1} + A_{1S}R_f\dot{h_1} + \frac{A_{1S}}{C_{fR}}h_1 = P_t \tag{15}$$

Equations (6), (13) and (14) describe the behavior of the water lock system, that is water equalization between Locks L1 and L2 during an outbound ship crossing. Along with Equation (15),

this set of equations fully represents the three sequences of behavior for the water lock fluid system during an outbound crossing.

4.2. System Behavior Modeling with Networks of Timed Automata

Timed systems are those where timing and scheduling of events (*i.e.*, logical results of computation) are relevant to correct operations [25]. In a hard real-time system, correct operation depends on the satisfaction of hard deadlines. Missing a deadline in a safety-critical system may have a disastrous effect. Soft real-time systems have correct operations that depend on the satisfaction of "average time" constraints.

Many complex engineering systems can be easily modeled using networks of communicating finite state machines (e.g., software systems; digital circuits; control of traffic through intersections). Individual processes are represented as finite state machines (FSM); embedded software systems can be modeled as networks of communicating FSM, as illustrated in Figure 5. A deterministic finite automaton (DFA) is a simple machine that reads an input string (one symbol at a time) and then, after the input has been completely read, decides whether to accept or reject the input [26]. In mathematical terms, a DFA is a five-tuple $(Q, \sum, \delta, q_o, F)$ where: (1) Qis a set of states; (2) $\sum$ is a finite alphabet; (3) δ: Q $\times$ $\sum$ $\rightarrow$ Q is the transition function; (4) $q_o \in$ Q is the start state; and (5) F $\subseteq$ Q is the set of accepting states. The language recognized by a DFA M is the set of all strings accepted by M and is denoted L(M). Simply put, automata are simple, but useful models of computation. They keep notions of state, stepping between states and well-defined start and end states, but omit notions of memory, variables, commands and expressions. Constructions on automata are a way of systematically building automata from (smaller automata) components [26]. In other words constructions operate on automata, yielding new automata. Therefore, constructions are a way of systematically building automata from (smaller automata) components. For example, the product automaton takes two DFAs and delivers the product automaton, also a DFA. Conceptually, the product automaton is a single machine that runs its two component machines in parallel on the same input string. At each step of the computation, both machines access the (same) current input symbol, but make transitions according to their respective transition functions. Construction operations provide closure under union, intersection, complement and concatenation operations.

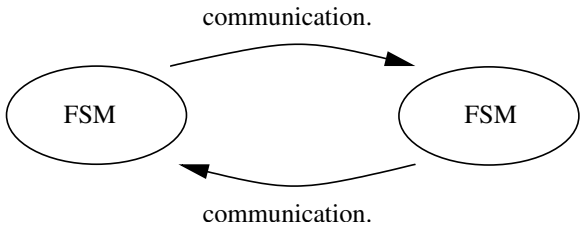

Figure 5. Modeling of systems as networks of communicating finite state machines (or more simply, as networks of communicating finite state automata).

A timed automaton is a stripped-down finite-state machine extended with clock variables. Mathematically, we state that a time automaton A over clocks C and actions Act is a four-tuple (L, l_o, E, I) where: (1) L is a finite set of locations; (2) l_o is the initial location; (3) E $\subseteq$ L $\times$ B(X) $\times$ Act $\times$ P(C) $\times$ L is the set of edges; and (4) I:L $\rightarrow$ B(X), each location to an invariant. Within an automaton model, the system's states are defined by the locations of all automata, the clock constraints and the values of discrete variables in the model. Every automaton may fire an edge separately or synchronize with another automaton, which leads to a new state. Edge firings are not allowed to be carried out simultaneously; therefore, the automata do not really operate in parallel, but rather in an interleaving way.

4.3. Model Checking

While semi-formal procedures such as SysML are useful for preliminary design (mainly because they force a designer to think about what they really want!), some errors, such as safety and liveness conditions, may be far too subtle to catch by inspection. Therefore, there is a strong need for automated procedures for the analysis of design specifications. Model checking [27] is a technique that relies on building a finite state model of a system and checking that a desired property holds in that model. Given a state-transition graph (M) and a formula (f), the model checking problem aims to decide whether the formula (f) is true for all possible runs. In mathematical terms, model checking solves the problem:

$$M, s| = f \tag{16}$$

That is, find all states "s" of "M", such that formula "f" holds. If the property does not hold, then a counter example will be provided. To date, model checking has been used primarily in hardware and protocol verification; the current trend is to apply this technique to analyzing specifications of software systems.

4.4. Model Checking with UPPAAL

UPPAAL [8] is an integrated tool environment for modeling, simulation and verification of finite-dimensional real-time systems that can be modeled as a collection of non-deterministic processes (timed automata). Assessment procedures are described in terms of temporal logic. Typical areas of application include real-time controllers and communication protocols in particular, those where timing aspects are critical. Thus, as a real-time system, our water lock system falls within the scope of this software.

System behavior is described in terms of timed automata extended with variables and real-valued clocks where clock variables evaluate to a real number, increase without bound and may be associated with a transition, in a guard or in a reset, and with a node, using an invariant. This is a dense-time model. The state of the system is defined by the locations of all automata, the clock constraints and the values of the discrete variables. Every automaton may fire an edge separately or synchronize with another automaton, which leads to a new state. Edge labels may include constraints on integer variables and array components in their guards, similar to timing constraints, and may specify assignments to variables in their actions, which are executed simultaneously.

To avoid the need for dealing with an infinite number of clock values, UPPAAL uses zones defined by sets of constraints on clock values. When a system is in a particular state (location), we do not have a concrete value of time; instead, differences defined by region boundaries. A particular simulation explores only a particular execution trace (*i.e.*, sequence of states of the system). The model checker uses state-space exploration to ascertain whether certain combinations of control nodes and constraints on clocks and integer variables are reachable from an initial configuration. Other properties, such as bounded liveness, can be checked by reasoning about the system in the context of testing automata or annotating the system description with debugging information and then checking reachability properties. In checking a property, a diagnostic trace can be automatically reported explaining why the property is satisfied or not. This trace can be visualized by running it through the simulator.

5. Case Study: Model-Based Design and Formal Validation of a Canal Water Lock

We now exercise the proposed methodology through the model-based design and formal validation of behaviors for a ship crossing a water lock by following the design flow illustrated in Figures 3 and 4. For the purposes of illustration, our main focus will be on the fluid subsystem of the water lock system. The primary function of a two-lock system is to safely carry ships from one water source to another in a timely manner. Lock chambers are the centerpieces and bottlenecks of the canal navigation systems; they define system capacity and performance and regulate its operations.

The computer system, which is overseen by operators, is responsible for ensuring that ship movements are safe and efficient.

5.1. Water Lock Operations and Fluid System Design

Fragments of water lock system behavior are motivated by scenarios, such as sequences of operations for inbound and/or outbound crossings and associated requirements to achieve acceptable levels of performance and safety. Inbound crossings occur when a ship enters the water lock from the sea and exits it on the lake side. Outbound crossings are defined by a ship traveling in the opposite direction. The case study will focus on an outbound crossing (*i.e.*, as a ship traverses from right to left in Figure 1), with the ship initially waiting in Lock 2 (L2). As already noted in the Introduction, the crossing is executed in a sequence of three fluid-flow operations:

Step 1. Water is transferred from Lock 2 to the reservoir (R), then,
Step 2. Water is transferred from the reservoir (R) to Lock 1 (L1) for equalization with L2.

Finally, after the ship is towed into L1:

Step 3. Water is transferred from L1 back to the reservoir R, thereby allowing the ship to be lowered to sea level, then towed out.

Thus, the fluid system behavior needs to satisfy three requirements:

1. Water equalization between locks during towing,
2. Fluid system operation *i.e.*, elevate and lower ships, and,
3. Limited reservoir capacity.

For equalization purpose, water in the chamber L1 has to be raised to the same level as in L2 to allow the tow boat to tow the ship from L2 to L1.

5.2. System Architecture and Modeling

The requirements engineering subprocess uncovered the following results:

- Six goals covering the desired functionality, performance, maintenance, security and economics of the waterway system operation. For convenience, the corresponding use-cases can be organized into four viewpoints: ship, humans, computer system and system engineer.
- Thirteen use-cases stemming from the six goals. The interactions between the water lock system and external actors in each use-case are described in more detail by a textual scenario.
- A use-case (UC) task interaction matrix and model.
- Scenario specifications with a step-by-step description of the expected functionality, with a traceability back to individual use-cases and individual goals.

The following snippet illustrates the pathway from a goal to a use-case and related textual scenarios.

Goal 1: The lock system must allow large container ships to traverse the lock system.
Use-case 01: Operate the lock system.
Scenario 1.1: A large container ship crosses the lock system inbound in a small amount of time.
Scenario 1.2: A large container ship crosses the lock system outbound in a small amount of time.
Scenario 1.3: The system accomplishes its operations with a minimal amount of water usage.

Use-case 01: Operate the lock system expresses the system functionality of receiving and processing a crossing request from a ship and is elaborated with Scenarios 1.1, 1.2 and 1.3, as they all involve some control of the crossing process. A complete scenario description (details not included here) will include preconditions for the scenario to activate a step-by-step description of required functionality under normal operations, and step-by-step descriptions for how the system will react in response to various types of failure. While the general relationship between goals and scenarios is many-to-many,

a one-to-one relationship exists between use-cases and their specification. There should only be one detailed description of each functionality of the system throughout the design flow. Analysis of abnormal events in use-case specifications along with preliminary behavior and structure designs provides a wealthy source of data and information for the identification of derived requirements that progressively expand and enrich the requirement tree, starting from the initial statement, as shown in Figure 6.

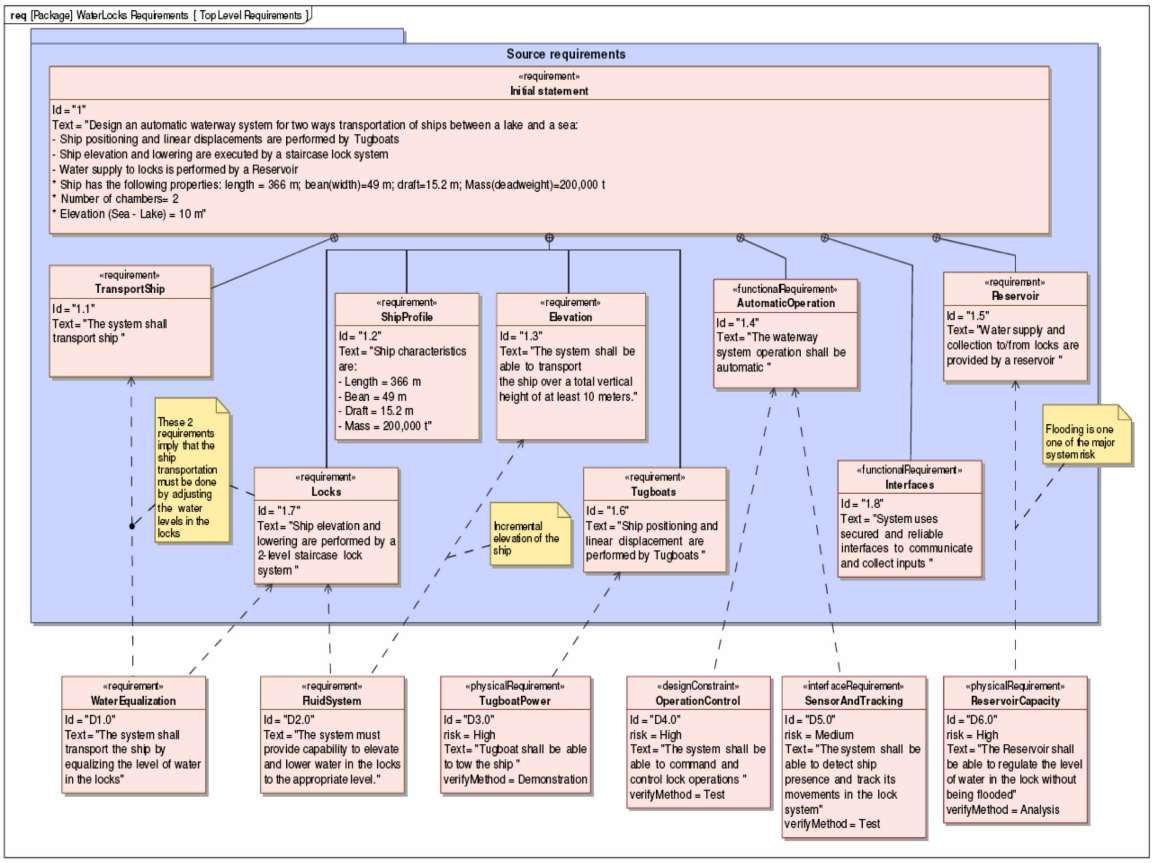

Figure 6. Water lock system requirements. Lower level requirements $D_{1.0}$, $D_{2.0}$ and $D_{6.0}$ have direct effects on the design of the fluid system. However, its operation is highly influenced by $D_{4.0}$ and $D_{5.0}$, as well as the constraint set by Req.1.2.

A second benefit of textual scenarios is the insight they provide toward figuring out the sequences of actions (or steps) for correct execution of system functionality. A careful examination of the different elements involved, considering the constraints defined by lower level requirements, helps uncover elements of the system structure and the sequences of message exchange needed for the implementation of system behavior. For the case study problem, this leads to the initial design where a controller, two pumps, three valves and a water reservoir are identified as the key elements of the fluid system. Figure 7 illustrates the system behavior and its allocation to structural elements of the fluid system (FS). The structural elements are as follows.

- FS controller: control and regulate the portion of crossing operations performed by FS,
- Lock (x2): provide room and a road (water) to the ship during crossing process,
- Valve (x3): drain water from/to the locks to/from the reservoir,
- Pump (x2): displace water from/to the locks to/from the reservoir to the required elevation and
- Reservoir: store reusable water for crossing operations.

We synthesize the design by combining the structural decomposition of the FS and its behavior. Figure 8 is an internal block diagram that shows: (1) how the various parts of the FS are connected to

one another; and (2) unidirectional flows (*i.e.*, water, signal, energy) that occur through input/output ports during equalization between components. We can also identify some of the water flows represented in Figure 7, for instance $Wf_1 \Longleftrightarrow$ Low1D2or $Wf_2 \Longleftrightarrow$ Low2D2.

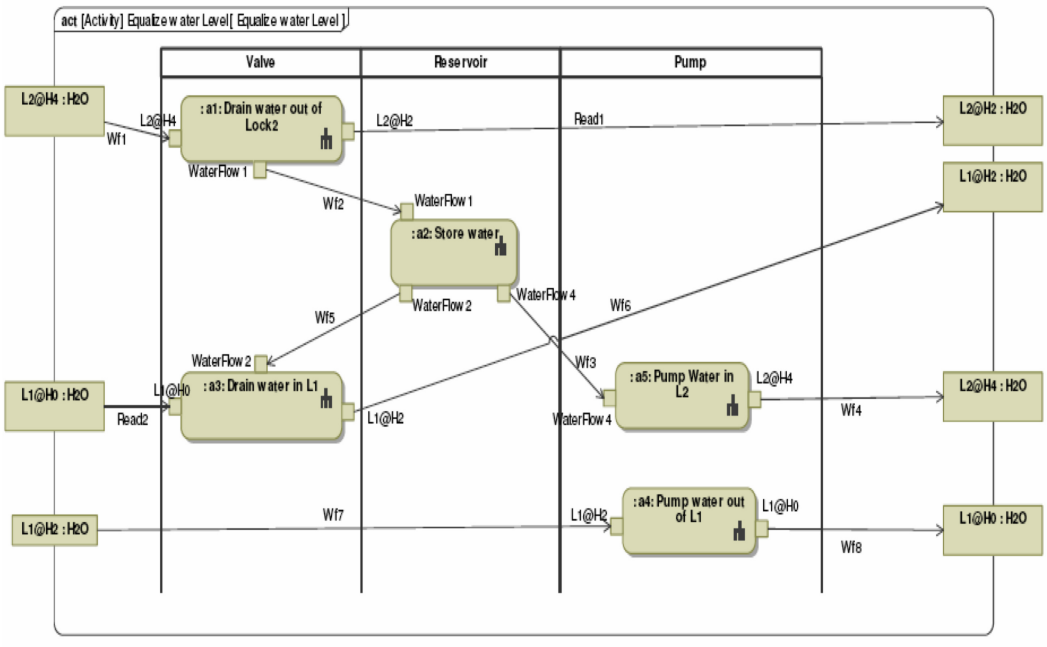

Figure 7. System Modeling Language (SysML) activity diagram for fluid-system behavior during the equalization of water levels. Behaviors are allocated to elements of the system structure (pump, valve and reservoir). The action of the controller on structural elements allows the water to change height (H_i) and to flow (Wf_j) in and between the lock (L_k) and reservoir containers.

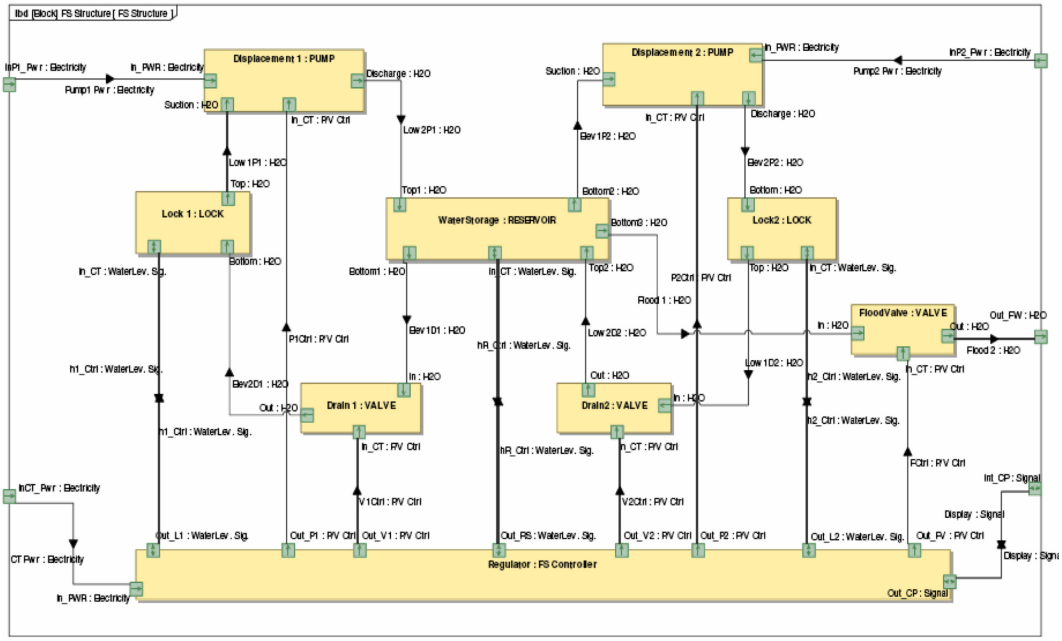

Figure 8. SysML internal block diagram (outbound crossing) of the synthesized fluid-system (FS) design. The FS controller is powered by electricity, and it controls structural components (valves, pump) through control signals sent to actuators. Sensors on the components send feedback signals to the FS controller and collect measurements.

It is important to keep in mind, however, that due to the weak semantics of the SysML modeling language, this process is not free of errors, ambiguities, redundancies and omissions [28]. As such, at this point, the model artifacts are not complete enough for performance and safety evaluation, as per the system requirements. To close this gap, we need to customize the mathematical equations to the sequence of fluid flows needed to allow a ship to descend through the lock system, build and simulate formal models of the system and compare the results to the expected levels of performance.

5.3. Time-History Simulation

We employ OpenModelica to solve the set of differential Equations (6) and (13) to (15) for an outbound crossing. The implementation incorporates constrained variables of the system requirements shown in Figure 6 and relevant predefined physical variables and parameters.

Figure 9 shows the result for each of the three steps introduced in Section 5.1. From Figures 1 and 6, and the textual requirements therein, the relevant values for the water level are as follows: (1) H_0 = 17 m is the sea water elevation at the foot of the lock structure; it is not 0 m, because there needs to be a minimum of a 15.2-m deep of water to account for the draft (see Req.1.2); (2) H_2 = 22 m is the equalized water elevation between Locks 1 and 2; it is half way between H_0 and H_4, as per Req. 1.3; and (3) H_4 = 27 m the elevation of the lake. We can verify that $H_4 - H_0$ = 10 m, as per Req. 1.3. The first plot, which lasts approximately 10.5 s, shows what is happening during Step 1. When the water height in Lock 2 decreases by five meters, we observe a linear increase of the water in the reservoir. The former is half of the specified value in source Requirement 1.3 (elevation). In the absence of a pump, Step 2 (middle plot) shows that it takes almost 12 h for the water to transfer from the reservoir to Lock 1 by gravity flow, to reach the target height (H_2 = 22 m) in Lock 1 and to equalize with Lock 2. Equalization of water levels in Locks 1 and 2 enables the ship to be towed to Lock 1. The extremely long transfer time by gravity flow alone highlights the need for a pump to improve the performance of the fluid-system behavior. With the assistance of a pump operating at 1 psi, Step 3 lasts approximatively 9 s. The water level in Lock 1 is lowered from H_2 = 22 m to H_0 = 17 m, thus lowering the ship to sea level where it can be towed away.

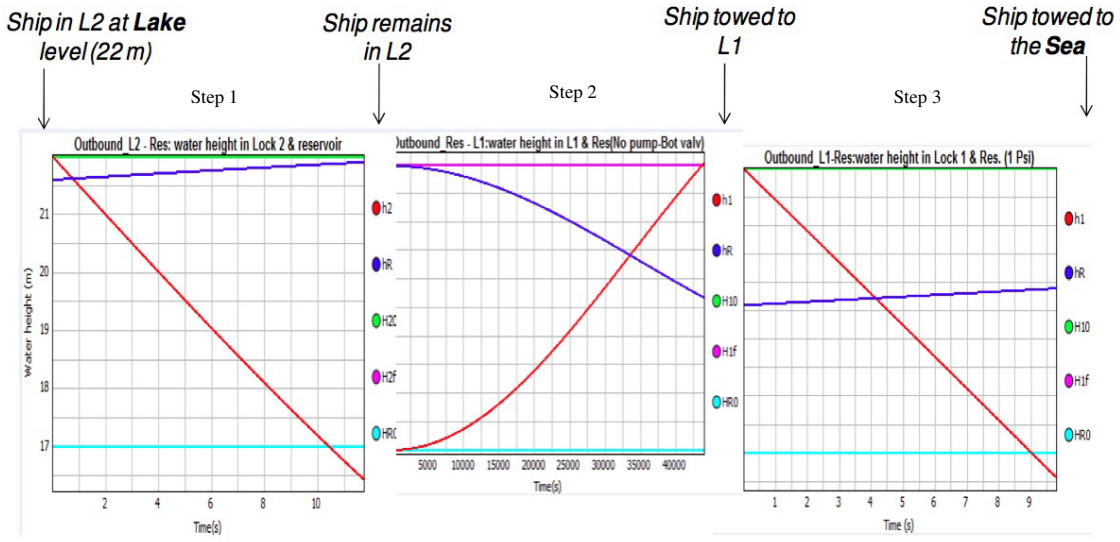

Figure 9. Simulation of the fluid-system behavior (*i.e.*, Step 1 → Step 2 → Step 3) during an outbound crossing. Step 3 is accomplished with a pump pressure = 1 ksi. Legend: h1 = variable for water level in Lock 1; h2 = variable for water level in Lock 2; hR = variable for water level in the reservoir; H10 = initial water level in Lock 1; H1f = final water level in Lock 1; HR0 = initial water level in the reservoir; H20 = initial water level in Lock 2; H2f = final water level in Lock 2.

A similar three-step process exists for the sequencing of fluid flows needed to elevate an incoming ship from sea level to lake level. Briefly, the details are: Step 1, transfer water from lock 2 to the reservoir; Step 2, elevate the ship by transferring water from reservoir to Lock 1; and Step 3, elevate the ship by transferring water from reservoir to Lock 2. In the preliminary simulation, the durations for each step are 10, 12,900 and 13,100 s, respectively.

5.4. Design Space Exploration

Design space exploration is the activity of systematically exploring design alternatives before implementation. Exploration can include determination of boundaries between the feasible and infeasible domains, as well as distributions of measures of effectiveness throughout the feasible domain. When the number of design alternatives is very large, exhaustive enumeration may be computationally prohibitive. From an end-user perspective, however, insight into the nature of the design space can be gained by systematically exploring small subsets of the design space. As a case in point, a designer might wonder if a pump is really needed to complete Step 3 of the outgoing traversal. Figure 10 shows the time history of water-level elevations for Step 3 when the pump is removed from the fluid-flow operations. Even though the pumping pressure (1 psi) is very small, it is nonetheless still needed. Similar analyses can be conducted to find a range of pumping pressures that will increase the flow rate and reduce the time needed to complete Step 2. A pump pressure of about 50 psi works well.

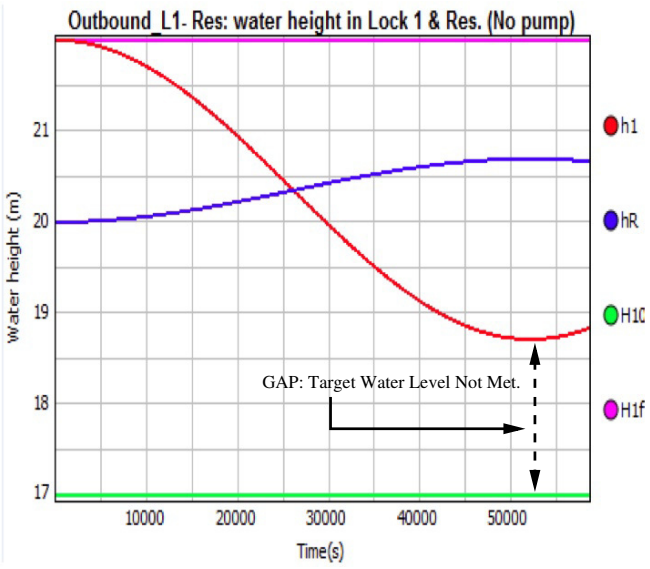

Figure 10. Design space exploration and an example of an infeasible design solution. Water elevation in the reservoir and Lock 1 (L1) *versus* time for gravity fluid flows (no pump). Legend: hr1 = variable for the water level in Lock 1; hR = variable for the water level in the reservoir; H10 = initial water level in Lock 1; H1f = final water level in Lock 1.

5.5. System Controller Design and Verification Approach

Now that the behavior associated with sequences of fluid-flows between the locks and the reservoir is in place, the next step is to consider simulation and verification of the control system design. It is important to note that while the fluid-flow behaviors are continuous, the lock system control is defined by sequences of discrete control actions acting over an ensemble of concurrent subsystem-level behaviors. We use timed automata models introduced in Section 4.2 to formally verify the correctness of operations for the waterway system controller and, thus, satisfaction of the safety constraints. The proposed approach takes the durations of time needed to complete the water transfer steps and uses them as input to the control model. Thus, the control systems verification problem boils down to sequencing of intervals of time and synchronization of events to achieve good system

performance, while assuring that the system operations remain safe. Such problems can be represented as networks of nondeterministic finite automata (NDFA) based on temporal logic and simulated and verified in UPPAAL (see Section 4.4).

Timing analysis: Before getting into the specifics of the controller design, it is necessary to identify and understand the interplay between the different processes that compose the system. Design structure matrices [29] provide a means to identify groups of tasks that should be clustered. For the inbound crossing process, twenty nine (29) low level tasks are bundled into five clusters corresponding to the following operations:

1. Transfer water from Lock 2 to the reservoir.
2. Transfer water from the reservoir to Lock 1.
3. Transfer water from the reservoir to Lock 2.
4. The ship leaves the lock system.
5. Reinitialize the lock system.

Figure 11 shows the interplay between time and the synchronization of events that occurs during an inbound crossing. The top row of Figure 11 shows the timing of actions for Steps 1 through 3. The middle row shows the timing of operations for Gates 0, 1 and 2. The bottom row shows the profile of water level elevation of the ship *versus* time. The vertical lines represent the actions of the system components during the ship traversal (e.g., ship enters Lock 1, Valve v2 opens, *etc.*). The ship, gate and valve operations communicate through message passing. At the conclusion of the ship traversal, the lock system is reinitialized so that it is ready to begin processing the next ship.

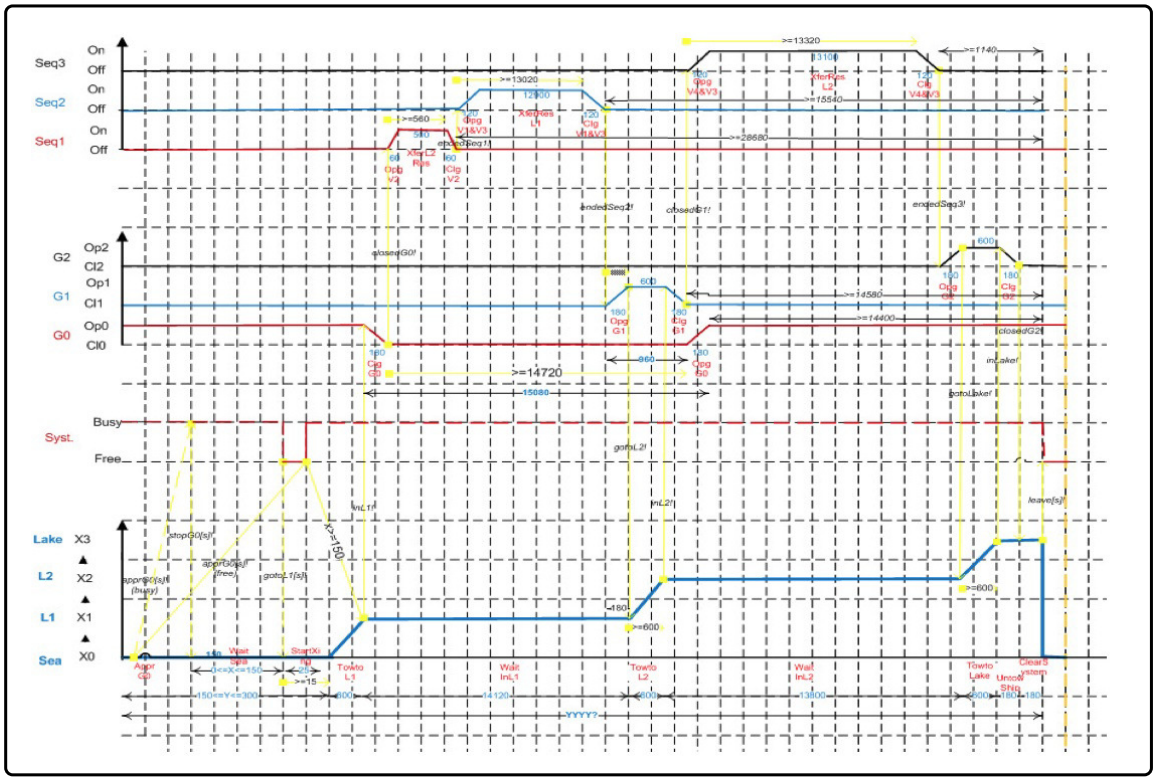

Figure 11. Timing diagram for an inbound water lock crossing.

5.6. Controller Modeling and Verification

Figure 12 shows the UPPAAL model of ship behavior for an inbound crossing of the lock system. The control model is composed of lower-level processes that participate in the sequence of actions that enable the crossing process. Based on the previous analysis, we have identified five components,

the behavior of which has been modeled in UPPAAL: ship, gate, valve, scheduler, main controller. The automata behaves as follows.

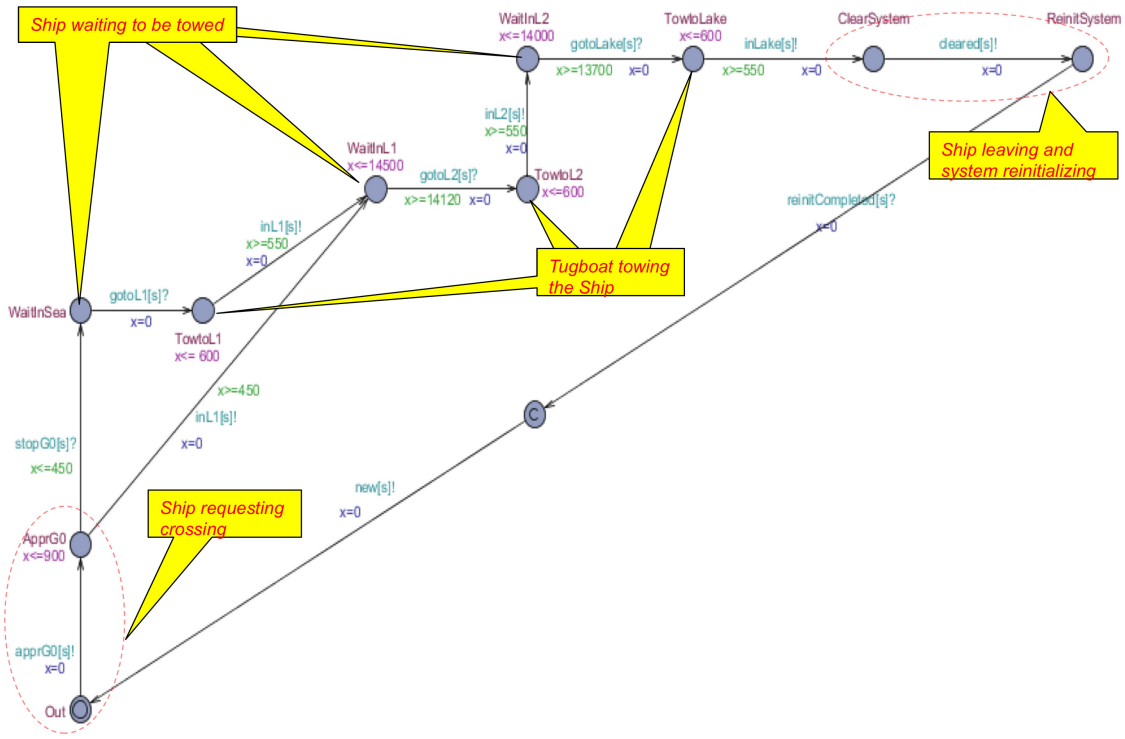

Figure 12. UPPAAL model of ship behavior during an inbound traversal of the lock system.

1. The ship approaches the lock system and requests crossing (automatic detection by a sensor). If the system is free, *i.e.*, there are no other ships in the water lock system, it will be towed directly into the lock ahead *i.e.*, Lock 1 for inbound crossing and Lock 2 for outbound crossing. Otherwise, it waits in the sea/lake for the signal from the controller to proceed.
2. The ship waits in the lock for the water to be raised/lowered up to the level of the water in lock ahead (equalization).
3. When the water in the actual lock is the same as the one in the lock ahead, the ship is towed to the lock ahead.
4. For inbound crossing, when the lake is reached, the ship is released and can leave the system, which has to be reinitialized before another ship can proceed.
5. For outbound crossing, the water in Lock 1 is lowered to the sea level, and the ship is towed in the sea and released; then, the system is reinitialized.

Similarly, the gates' and valves' behaviors are modeled as a network of automata with open/opening and closed/closing as basic states. The execution of the states is constrained by their timing as per the simulation results in Section 5.3 and the timing analysis in Section 5.5. Furthermore, a scheduler is needed to manage the queue at the entrance of the lock system. It determines the order in which the ships will cross the system and communicates the appropriate information to the controller and the incoming or waiting ships. It also encapsulate the global status of the water lock system through its state, which is either busy or free.

Main controller: The main controller of the waterway system is the element at the center of the waterway operations. The controller interacts with each of the aforementioned components, sending signals and messages and synchronizing operations to achieve a proper execution of each task in real time. This leads to a complex set of tasks performed in a timely and safe manner.

Figure 13 shows the behavior model for our controller. Controlled behavior can be summarized by the following macro steps (highlighted by the call outs):

1. After the scheduler frees a space in the queue, the ship requesting the crossing gets into Lock 1, and the controller oversees the Hate 0 closing process.
2. The controller sends the proper signal (opening and closing) to the valve actuators in order to start Sequences 1 and 2, which correspond to water equalization between Lock 1 and 2. It controls the opening and closing of valves, as well as the durations of sequences.
3. Once Step 2 is completed, it opens Gate 1 to let the ship get into Lock 2 and also performs the closing of Gate 0.
4. It now commands actuators of valves V_3 and V_4 to perform Step 3 (water equalization between Lock 2 and the lake). It controls both the time and proper execution of the sequence.
5. Having reached the lake, the ship can now exit the system. The controller first ensures the Gate 2 is closed and oversees the ship clearing the water lock system.
6. In the last step, the controller reinitializes the system by activation of reservoir and Lock 1 flooding valves to bring the water in those containers at the appropriate level prior to the beginning of the next crossing cycle.

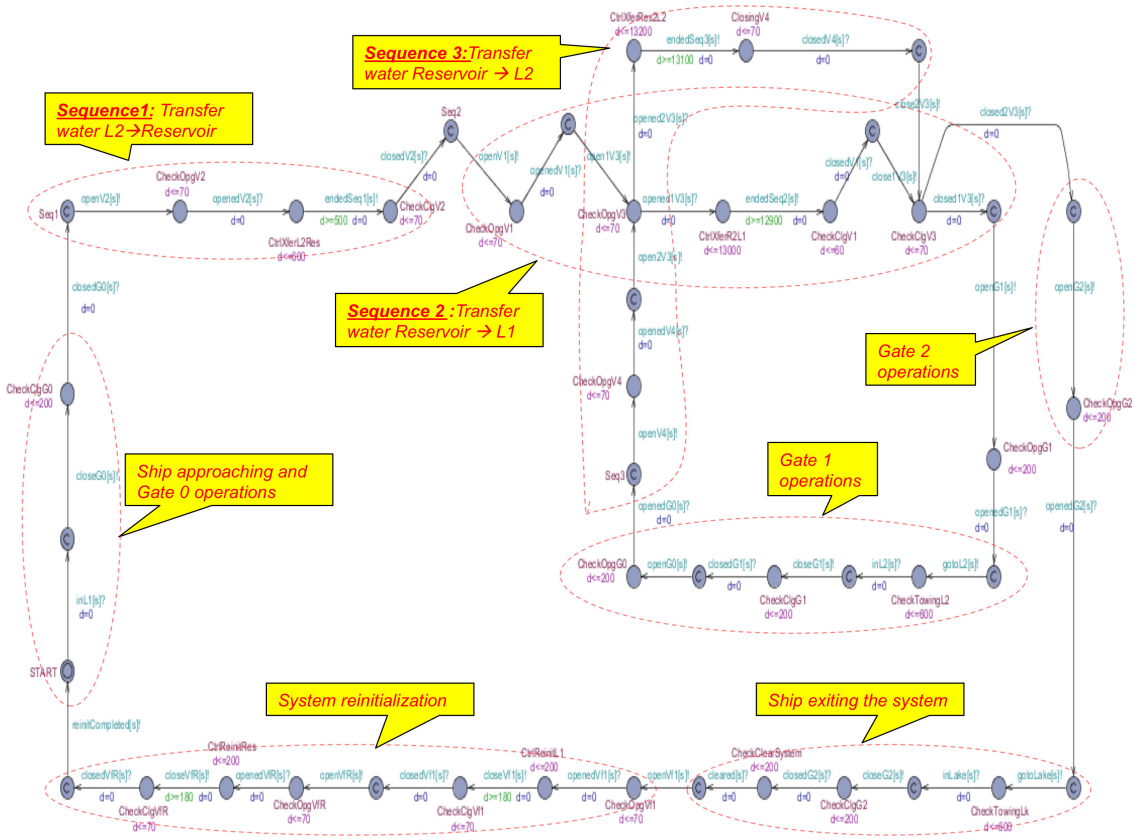

Figure 13. Simplified UPPAAL model of the system controller.

5.7. Model Simulation and Verification

Simulation trace: The proper composition of the various processes modeled above renders the simulation of the behavior of the fluid system possible in UPPAAL. During the simulation, we can visualize and extract the trace of the crossing process execution for further analysis. Figure 14 shows a view of the simulation of the system with its trace. In this case, three ships are requesting the crossing of the water lock system, and only one (Ship 1) is actually granted that right, while others are waiting in the sea; this is actually the kind of behavior expected from our system.

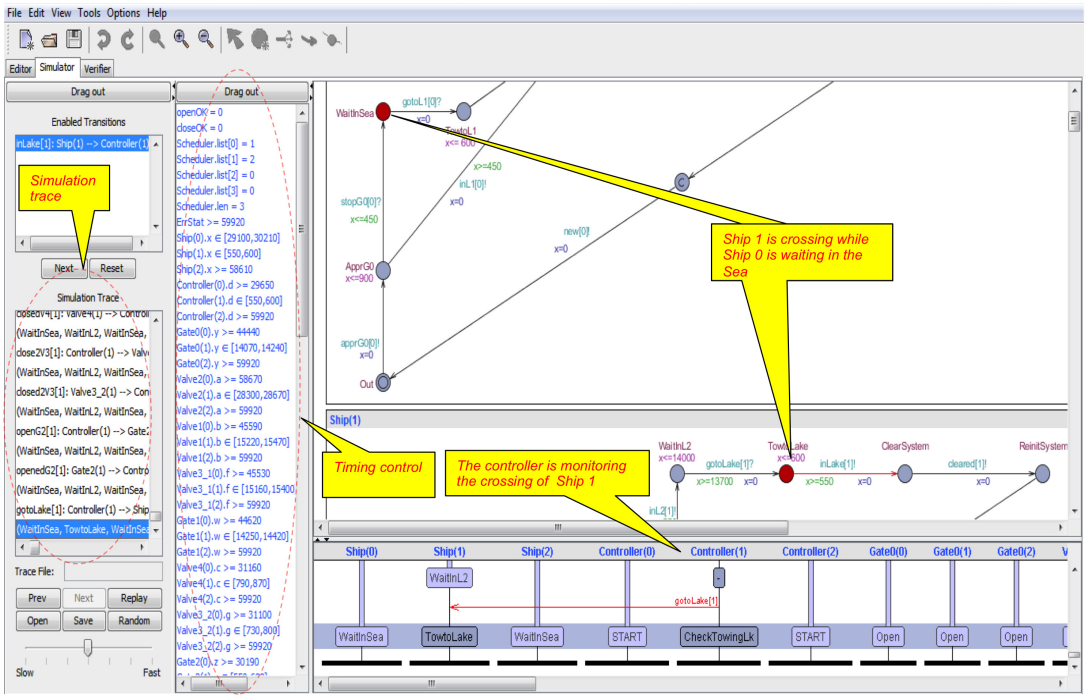

Figure 14. Model simulation and the view of a simulation trace.

Verification and system traceability matrices: The simulation of the system provides us a glimpse of the proper execution of the crossing process. However, only a formal verification of the system requirements can provide a definitive proof of the correctness of the design and system operation. Thus, we use UPPAAL Verifier to specify the queries formally verifying the system properties. Queries are organized into three categories of properties, as shown in the third column of Table 1:

Safety: A safety property asserts that nothing bad happens, such as an accident, to the system during its operation. Properties P_1 through P_3 are illustrations.

Liveness: A liveness property asserts that, for a given state of the system, it will eventually progress onto the next one *i.e.*, something good will eventually happen. Thus, it requires the water lock system (automated and distributed) to make progress despite the fact that some of its concurrent processes may have to run in sequence at time, especially on parts of the composed system that cannot be simultaneously ran by multiple processes. Properties P_4 through P_{11} are examples of liveness properties.

Reachability: A reachability property asserts that, given an initial state s_0, there exists a path p that allows the system to reach a particular state s_f in the future. Such properties are P_{12} to P_{15}.

Table 1 summarizes the results of system properties verification. For the case (N = 3 ships), all of the key safety and reachability properties are verified. Of the three properties not verified, the first (P_4) is due to a design choice we made (no duplication of V_3 representation to simplify the model of the controller). However, property P_5 ensures that that choice does not impact the ability of the system to operate properly. The second (P_9) can be explained by the fact that its satisfaction depends on the beginning of a new cycle, constrained by the presence or not of a ship requesting crossing, which is not always guaranteed. Given the complexity and volume of verification tasks needed to ensure its correctness, the last one (P_{10}) is much more complex to troubleshoot. We suspect, however, the reason behind the non-verification of P_4 to be the cause of this failure. Our belief is reinforced by the fact that P_{11} ensures us of the safe operation of the system the same way P_5 did for P_4. Thus, we can conclude from these results the proof of the correctness of our controller in satisfying automatic water lock operations.

Table 1. Summary of the simulation parameters (legend: S = safety; L = liveliness; R = reachability).

ID	Property	Type	Query (UPPAAL)	Satisfied?
P_1	The system is deadlock free.	S	$A[]$ not deadlock	Yes
P_2	There is never more than one ship crossing the lock.	S	$A[]$ forall (i : id_s) forall (j : id_s) Ship(i).TowtoL2 && Ship(j).TowtoL2 imply i == j	Yes
P_3	There can never be N ships (>1) in the queue (thus the array will not overflow).	S	$A[]$ Scheduler.list[N] == 0	Yes
P_4	Whenever a ship approaches the system, it will eventually cross.	L	Ship(0).ApprG0 $\rightarrow$ Ship(0).TowtoL2	No
P_5	A ship approaching the system will ultimately find a way to cross.	L	$E[]$ Ship(0).ApprG0 imply Ship(0).TowtoL2	Yes
P_6	Whenever a gate opens, it will eventually close later.	L	Gate1(1).Open $\rightarrow$ Gate1(1).Closed	Yes
P_7	Whenever a gate closes, it will eventually open later.	L	Gate1(1).Close $\rightarrow$ Gate1(1).Opened	Yes
P_8	Whenever a sequence start, it will finish at a certain point in time.	L	Controller(0).Seq2Start $\rightarrow$ Controller(0).Seq2End	Yes
P_9	Whenever a sequence finish, it will start over at a certain point in time.	L	Controller(1).Seq2End $\rightarrow$ Controller(1).Seq2Start	No
P_{10}	Whenever the system is busy, will system free eventually hold?.	L	Scheduler.Busy $\rightarrow$ Scheduler.Free	No
P_{11}	A busy system will ultimately become free.	L	$E[]$ Scheduler.Busy imply Scheduler.Free	Yes
P_{12}	Any ship can reach at least Lock 2.	R	$E <>$ Ship(1).WaitInL2	Yes
P_{13}	Any ship crossing inbound can reach the lake.	R	$E <>$ Ship(0).ClearSystem	Yes
P_{14}	Ship 0 can be crossing the lock system, while Ship 1 is waiting to cross.	R	$E <>$ Ship(0).TowtoL2 and Ship(1).WaitInSea	Yes
P_{15}	Ship 0 can cross the lock system while the other ships are waiting to cross.	R	$E <>$ Ship(0).TowtoL2 and (forall (i : id_s) i ! $= 0$ imply Ship(i).WaitInSea)	Yes

Table 2. System traceability matrix (partial).

Use Cases		Source Requirements		Derived Requirements		Fluid System	
ID	**Name**	**ID**	**Name**	**ID**	**Name**	**Spec.**	**Components**
UC_{01}	Operates the lock system	1.4	Automation	$D_{4.0}$, $D_{5.0}$	Operation control; sensors and tracking		Controller
		1.3	Elevation	$D_{2.0}$	Fluid system	UR1-5, VR1-6, PR1-5, RR1-5	Pump, valve, pipe, reservoir
UC_{02}	Provides water to locks	1.5	Reservoir	$D_{6.0}$	Reservoir capacity	VR1-6; PR1-5; RR1-5	Reservoir, pipe, valve
		1.1	Transportation	$D_{1.0}$	Water equalization	LR1-4; UR1-5; VR1-6; PR1-5; RR1-5	Lock, reservoir, pump, valve, pipe
UC_{03}	Provides system health information	1.8	Interfaces	N/A			
UC_{04}	Logs into the system	1.8	Interfaces	N/A			
UC_{05}	Monitors lock operations	1.7	Locks	$D_{1.0}$, $D_{2.0}$	Water equalization; fluid system	LR1-4; UR1-5; VR1-6; PR1-5; RR1-5	Lock, reservoir, pump, valve, pipe
		1.8	Interfaces	N/A			
UC_{06}	Navigates lock system	1.2	Ship profile	N/A			
		1.6	Tugboats	$D_{3.0}$	Tugboat Power		
		1.8	Interfaces	N/A			
UC_{07}	Stores reusable water	1.5	Reservoir	$D6.0$	Reservoir capacity	RR1-5	Reservoir

System traceability: At this point, we can trace the system initial use-cases to the component specifications, as well as the mechanisms through which they are related to system functionality. From this perspective, the derived system requirements serve as bridges between source requirements and subsystem components requirements, *i.e.*, the fluid system in this case. We can also observe that not all source or derived requirements are tracked down to the components of the fluid system. This actually makes sense since as many as seven other subsystems need to be developed in order to obtain a complete design of the water lock system (see Figure 1). Table 2 provides a convenient means of ensuring that all of the system requirements associated with the fluid system are satisfied by the selected components. Thus, we can check the box for Requirement 1.4 (under reserve on the final testing of requirements $D_{4.0}$ and $D_{5.0}$ once the system is built).

6. Discussion and Conclusions

Waterway systems are cost effective in the transport of bulk and containerized goods to support global trade. Yet, they are among the most under-appreciated forms of transportation engineering systems. Looking ahead, the long-term view is not rosy. Failures, delays, incidents and accidents in aging waterway systems are doing little to attract the technical and economic assistance required for modernization and sustainability. In a step toward overcoming these challenges, the purpose of this paper has been to describe a multi-level multi-stage methodology for the model-based design, simulation and formal verification of automated waterway operations. Challenges include the need to work with mixtures of informal and formal model representations, mixtures of continuous (fluid flows) and discrete (control actions) behavior and system-level behavior defined by the composition of multiple lower-level concurrent behaviors.

To illustrate the essential features of the proposed methodology, the case study example employed SysML for high-level representations of system structure and behavior, OpenModelica for the simulation of continuous fluid flow behaviors and UPPAAL for the composition and formal verification of the control system design. We have manually connected the various steps in the proposed methodology, but moving forward, some of the model transformations could be automated and the models organized into a design platform architecture [21]. For example, MagicDraw [30] can be used for the generation of SysML diagrams and connected to OpenModelica (and Jython [31] and MATLAB) through plugins. Co-simulation of concurrent behaviors is also possible through the use of the Functional Mockup Interface (FMI) standard [32]. In either case, aggregated results of the OpenModelica simulations show up as parameter values in blocks of SysML parametric diagrams. Work is also underway to automate the pathway from simulation to model checking with adaptive interval abstractions [33].

Acknowledgments: This paper began as project reports prepared by the first author for ENSE 622 Systems Engineering Requirements, Design, and Trade-Off Analysis, and ENSE 623 Systems Engineering Design Projects, Validation and Verification, graduate courses in the Master of Science in Systems Engineering Program, Institute for Systems Research, University of Maryland, College Park.

Author Contributions: Leonard Petnga conceived, designed and solved the case study; Mark Austin developed the model-based design and verification pathways. Both authors wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Grier, D.V. *The Declining Reliability of the U.S. Inland Waterway System*; U.S. Army Corps of Engineers, Institute for Water Resources: Alexandria, VA, USA, 2005.
2. Jackson, D.E., Jr. *Leveraging the Strategic Value of the U.S. Inland Waterway System, USAWC Strategy Research Project*; US Army War College, Carlisle Barracks: Carlisle, PA, USA, 2007.
3. Iowa Department of Transportation. *U.S. Inland Waterway Modernization: A Reconnaissance Study*; HDR Engineering Inc.: Ames, IA, USA, 2013.
4. Schexnayder, C.J. Firms Eye Billions in Expansion Work At Panama Canal. Available online: http://panama-guide.com/ (accessed on 1 June 2016).

5. James, T. All Locked Up. *Comput. Control Eng.* **2006**, *17*, 16–21.
6. Kaisar, E.; Austin, M.A.; Papadimitriou, S. Formal Development and Evaluation of Narrow Passageway System Operations. *Eur. Transp.* **2006**, *34*, 88–104.
7. Kaisar, E.; Austin, M.A. Synthesis and Validation of High-Level Behavior Models for Narrow Waterway Management Systems. *J. Comput. Civ. Eng. ASCE* **2007**, *21*, 373–378.
8. UPPAAL: An integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata. 2004. Available online: http://www.uppaal.com/ (accessed on 1 June 2016).
9. SysML Plugin, No Magic Inc. Available online: http://www.nomagic.com/ products/ magicdraw-addons/ sysml-plugin.html (accessed 30 January 2014).
10. Fritzson, P. *Principles of Object-Oriented Modeling and Simulation with Modelical 2.1*; John Wiley and Sons: Hoboken, NJ, USA, 2003.
11. Gribar, J.C.; Bocanegro, J.A. Passage to 2000: Modernization of the Panama Canal. *Civ. Eng. Mag.* **1999**, *69*, 48–53.
12. Moore, M. The Bosporus: A Clogged Artery. The Washington Post. Available online: http://www.washingtonpost.com/ (accessed on 1 June 2016).
13. Wieringa, R. *Design Methods for Reactive Systems: Yourdon, Statemate, and the UML*; Morgan Kaufmann Publishers: San Francisco, CA, USA, 2003; p. 457.
14. Dai, M.D.; Schonfeld, P. Metamodels for Estimating Waterway Delays Through a Series of Queues. *Transport. Res.* **1998**, *32*, 1–19.
15. DeSalvo, J.S.; Lave, L.B. An Analysis of Towboat Delays. *J. Transp. Econ. Policy* **1968**, *2*, 232–241.
16. Ting, C.J.; Schonfeld, P. Integrated Control For Series of Waterway Locks. *ASCE J. Waterw. Port Coast. Ocean Eng.* **1998**, *124*, 199–206.
17. Zhu, L.; Schonfeld, P.; Kim, Y.; Flood, I.; Ting, C.J. Queuing Network Analysis for Waterways with Artificial Neural Networks. *Artif. Intell. Eng. Des. Anal. Manuf.* **1998**, *13*, 365–275.
18. Austin, M.A.; Baras, J.S.; Kositsyna, N.I. Combined Research and Curriculum Development in Information-Centric Systems Engineering. In Proceedings of the Twelth Annual International Symposium of the International Council on Systems Engineering (INCOSE), Las Vegas, NV, USA, 28 July–1 August 2003.
19. Austin, M.A.; Mayank, V.; Shmunis, N. PaladinRM: Graph-Based Visualization of Requirements Organized for Team-Based Design. *Syst. Eng.* **2006**, *9*, 129–145.
20. Jackson, D. Dependable Software by Design. *Sci. Am.* **2006**, *294*, 68–75.
21. Mosteller, M.; Austin, M.A.; Yang, S.; Ghodssi, R. Platforms for Engineering Experimental Biomedical Systems. *IEEE Syst. J.* **2014**, *9*, 1–11.
22. Fridenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML*; The MK-OMG Press: Burlington, MA, USA, 2008.
23. Unified Modeling Language (UML). 2003. Available online: http://www.omg.org/uml (accessed on 1 June 2016).
24. Wellstead, P.E. *Introduction to Physical System Modelling. Control Systems Principles*; Academic Press Ltd: London, UK, 2000; p. 256.
25. Hooman, J. *Introduction to Timed Systems*; University of Nijmegen: Nijmegen, The Netherlands, 2001.
26. Slind, K. *Class Notes on Theory of Computation*; Department of Computer Science, University of Utah: Salt Lake City, UT, USA, 2004.
27. Baier, C.; Katoen, J.P. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008.
28. Graves, H. Integrating Reasoning with SysML. In Proceedings of the INCOSE International Symposium, Rome, Italy, 9–12 July 2012.
29. Eppinger, S.D.; Browning, T.D. *Design Structure Matrix Methods and Applications*; The MIT Press: Cambridge, MA, USA, 2012.
30. MagicDraw. 2016. Available online: http://www.nomagic.com/products/magicdraw.html (accessed on 4 May 2016).
31. The Jython Project. 2016. Available online: http://www.jython.org (accessed on 4 May 2016).
32. Functional Mock-up Interface Standard for Model Exchange and Tool Coupling. 2016. Available online: https://www.fmi-standard.org (accessed on 4 May 2016).

33. Brauer, J.; Moller, O. Techniques for Abstraction of Continuous-Time Models into Finite Discrete-Event Models for Model Checking. Technical Note D5.1c, Version: 0.3, Horizon 2020: Integrated Tool Chain for Model-Based Design of CPSs. 2015. Available online: http://into-cps.au.dk (accessed on 1 June 2016).