Integrated Clinical Environment Device Model: Stakeholders and High Level Requirements

Yu Jin Kim, John Hatcliff, Venkatesh-Prasad Ranganath, and Robby[*]
Kansas State University

Sandy Weininger
US Food and Drug Administration

ABSTRACT

Both researchers and industry vendors are exploring the concept of systems of interoperable medical devices and health information systems that support safety and enhance clinical functionality. The Integrated Clinical Environment (ICE) has been proposed as an architecture for such systems, and several research and commercial implementations are being developed. Central to ICE concept of safe and flexible plug-and-play interoperability is the notion of an ICE Device Model (DM), which is intended to be a machine-readable meta-data description of a device's capabilities as exposed in standardized form over its network interface.

In this paper, we present an overview of the stakeholders and high level goals of the ICE DM as necessary for supporting safe and flexible plug-and-play interoperability. These goals provide the foundation for further refinement by research and industry teams to specific safety and implementation requirements for the ICE DM concept.

1. INTRODUCTION

From a systems engineering perspective, it is well understood that the current state of medical practice often uses *non-integrated* devices and health information systems (HIS) according to *informal manual protocols* to deliver clinical solutions. Integrating individual devices to support streaming of data to the Electronic Health Record (EHR) via Integrating the Healthcare Enterprise (IHE) [2] and Health Level 7 (HL7) [1] technologies has been a major emphasis in health technology community. However, there is even greater utility in safely integrating *collections of heterogeneous medical devices to work together as a system.* This requires architectural approaches and safety-critical technology solutions beyond what IHE and HL7 offer (though IHE and HL7 would likely play a role in the solutions). Providing the IT infrastructure, safety/security framework, and trustworthy compositional techniques to integrate devices and HIS and automatically coordinate their actions as a system of systems (SoS) would provide opportunities to, for example, implement multi-device smart alarms and safety interlocks, enable better clinical decision support, automate clinical workflows, and implement closed loop control.

A vision is emerging of a new paradigm of medical system enabled by *medical application platforms* (MAPs) [4]. A MAP is a safety- and security-critical real-time computing platform for: (a) integrating heterogeneous devices, medical IT systems, and information displays via a communication infrastructure, and (b) hosting application programs (i.e., *apps*) that provide medical utility via the ability to both acquire information from and control integrated devices, IT systems, and displays.

One can imagine many different types of MAP applications, but one common theme is that they introduce a previously missing "system perspective" (cross-device) into the device context associated with patient care. As an example of a multi-device safety interlock app, consider patients receiving an opioids (e.g., morphine) through a patient-controlled analgesic pumps that run the risk of overdose. An app can receive physiological data from a pulse oximeter and respiratory rate monitor that watch for signs of respiratory depression; if depression is detected, it can disable the pump to "lock out" the possibility of overdose [9].

Different architectural solutions for medical application platforms are possible. One architecture that is gaining traction is the Integrated Clinical Environment (ICE) [3] whose development has been led by the CIMIT Medical Device Plug-and-Play interoperability project and standardized in the ASTM F2761-2009 standard. F2761 defines the principle components of a MAP architecture and provides a rationale for the role that each component plays in establishing interoperability and safety. ASTM F2761 is now a FDA recognized standard.

A central element of the ICE vision is the *ICE Device Model* (ICE DM), which provides the foundation for safe and secure interoperability between ICE apps and devices. For a given ICE-compatible device, the device's ICE DM provides a declarative and machine-readable metadata description of the capabilities of the device (e.g., physiological parameters, alerts, settings, operations) exposed over the device's network interface – the ICE Equipment Interface.

The envisioned ICE DM, which is described only minimally in ASTM F2761, is somewhat analogous to the 11073

[*] This work was supported in part by the US National Science Foundation (NSF) awards CNS-1238431, CNS 1224007, CNS-1355778, CNS-1446544, and by CIMIT/Massachusetts General Hospital as a subcontract of a NIH/NIBIB Quantum grant. Copyright retained by the authors.

Domain Information Model (DIM) [7][8], in that they both provide a declarative description of device capabilities, and they are exchanged with a manager at association time to form a basis for interoperability. However, our vision of the ICE DM, associated tooling, and ecosphere, which we refer to collectively as the ICE DM framework, requires a number of features beyond the 11073 DIM to achieve the ICE goal of safe, secure, and flexible app-device interoperability. Additional requirements include features for safety, security, behavioral specifications, more flexible notions of compatibility, automatic conformance checking, automated code generation and testing, and more modern approaches to data representations and feature modeling.

Various organizations are prototyping ICE implementations, and professional organizations such as Association for the Advancement of Medical Instrumentation (AAMI) and Underwriters Laboratories (UL) are developing standards that address ICE. Regulatory agencies including the US Food and Drug Administration have an interest in tracking these developments to understand how emerging notions of an ICE DM will address safety and support risk management. To facilate a broader conversation about development and future standardization of an ICE DM, this paper presents our organization's view of the high-level requirements for the ICE DM driven by a presentation of stakeholder needs.[1]

These requirements provide guidance to research teams (including our own) and industrial vendors of interoperability solutions that may be interested in supporting the ICE architecture. Specific realization of ICE DM might come in the form of an evolution of the 11073 DIM or in a substantially new model framework (which might still leverage existing elements of 11073, such as the 11073 nomenclature[6]). While we phrase our presentation in terms of the ICE architecture, the concepts that we introduce are in many cases applicable to other possible MAP architectures.

2. ICE AND MDCF OVERVIEW

The ICE standard (ASTM F2761-2009 [3]) defines one particular architecture for MAPs. The boxes with dashed lines in Figure 1 present the ICE architecture. ASTM F2761 identifies an abstract "functional model" that includes components such the Supervisor, Network Controller, etc. with brief high-level description of the role of these components within the architecture. Future implementation standards are envisioned that provide detail implementation requirements and interface specifications for these components.

The ICE *Network Controller* provides a high-assurance network communication capability, establishing virtual "information pipes" between heterogenous devices (devices of different categories, often from different vendors) and apps running in the Supervisor. ASTM F2761 states that the ICE DM is a "representation of the capabilities of [a medical device] that includes information needed to qualitatively and quantitatively describe, control, and monitor its operation" and that the Network Controller "shall provide association to and communication with each attached [device] by interpreting the device model." Thus, the Network Controller exposes the ICE Interfaces of attached devices specified using the ICE DM to Supervisor apps. Because it establishes virtual information channels by *interpreting* the device models of the attached devices, it provides transportation for any information that can be described by a device model, and it is agnostic to the clinical intended use of data/control that it transports.

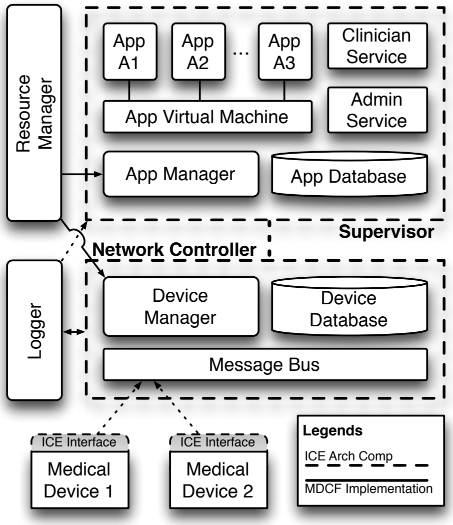

Figure 1: Decomposition of MDCF MAP services along ICE architectural boundaries

ASTM F2761 states that the *Supervisor* "provides a platform for functional integration between ICE compliant equipment via the network controller and can provide application logic and an operator interface" [3], but it does not explicitly identify: (i) the notion of an "app", nor (ii) the mechanism through which the "application logic" can be programmed, organized, or supported by an execution environment that provides appropriate guarantees.

MDCF: To flesh out requirements and architectural principles for these components, researchers from Kansas State (including the authors on this paper) and University of Pennsylvania have developed a prototype implementation for ICE called the Medical Device Coordination Framework (MDCF) (see, e.g., [11]). Figure 1 presents some of the primary MDCF components (drawn with solid lines) that fill out the ICE architecture.

In MDCF, a *Message Bus* provides Network Controller functionality via a high-level publish/subscribe messaging service; all communication between medical devices and the MDCF occurs via the Message Bus, including protocol control messages, patient physiological data, and commands. It also provides basic real-time guarantees (e.g., bounded end-to-end message transmission delays) that apps can state as requirements [10]. It provides notifications to the app when real-time constraints are violated to enable the app to take mitigating actions, and it also provides various fine-grained per-pipe role-based access control security policies. The DM-based description of a device's capabilities is used in multiple ways to provide automatic support for implementation and use of the device network interface. During device development, the MDCF development environment automatically generates APIs (Label 1 in Figure 2) that a de-

[1] There are several organizations working on developing ICE. The views expressed in this paper represent the views of our particular research team (based on a long history of work in language technology, publish-subscribe middleware and model driven development techniques) and are not to be interpreted as a definitive consensus opinion from the broader ICE community.

vice implements (Label 2) to provide the services implied by the DM description. At run time (upon device association), the MCDF Resource Manager automatically allocates communication channels (Label 3) (e.g., topics in the CORBA Data Distribution Service) to support communication associated with the physiological parameters, alerts, and device settings/operations specified in the device's DM. The Resource Manager also uses information in the DM regarding size of data packets, frequency of device publishing of data, and other real-time constraints to automatically manage quality of service and network scheduling provided by the Message Bus.

In the MDCF, the *Device Manager* implements device association functions (including verifying digital credentials attesting to a device's identity, ICE compliance, and regulatory clearance), stores the ICE DM received from each attached device during association, and tracks the connectivity of associated devices (Label 4).

The MDCF views the Supervisor as a virtual machine (VM) that hosts *Supervisor Apps.* The envisioned VM provides separation/isolation kernel-like [14] data partitioning (e.g., preventing information leaks between apps, and apps cannot inadvertently interfere with one another) and time partitioning (e.g., real-time scheduling guarantees that one app's computation or memory requirements cannot catastrophically affect the performance of another app).

Each app uses the DM language to specify the device capabilities that it requires to carry out its clinical function (Label 5). This DM-based approach is important for supporting a crucial integration function provided by the MDCF run-time environment; namely, when an operator requests the launch of an app, the MDCF will automatically check that the capabilities specified by a device's DM are compatible with the app's requirements (Label 6). Since both the app's requirements on devices as well as devices' provided services are specified in the DM language, this automated compatibility check is made significantly easier. In addition, during app development, the same DM-to-API code generation (Label 7) used for devices is used to generate APIs that the app employs to access device capabilities. Uniform and automatic generation of APIs (that behind the scenes utilize the underlying message bus) on both the "service side" (i.e., the device) as well as the "client side" (i.e., the app) leads to a fundamental property: *compatibility* between an app and devices on the DM level guarantees *interoperability* (Label 8) between an app and devices at the code/middleware level.

The DM framework and associated platform-provided compatibility check between apps and devices supports another notable goal of ICE: enabling a paradigm of *compositional compliance and safety reviews*. In this paradigm, devices can be assured to be compliant with their ICE DM specifications and can receive approval for use within ICE independent of a specific application context. Apps can be assured to satisfy their overall functional and safety requirements while relying not on the properties of specific devices but on its DM-based specification of device functional and performance properties – the app/device DM compatibility check ensures that the app will only operate with specific devices that satisfy those properties. Since the app's assurance of system function and safety relies only on its DM-based characterization of devices, the properties captured in the DM should be strong enough to support such assurance arguments. In particular, the DM should also support specification of *behavioral* properties of devices that characterize the devices intended function and performance, as well as means of specifying common fault situations. Further research and validation are neeeded to determine the *extent* to which a compositional paradigm will provide adequate levels of system assurace, and the identification of some of the high-level requirements presented in this paper is a necessary step in progressing the research.

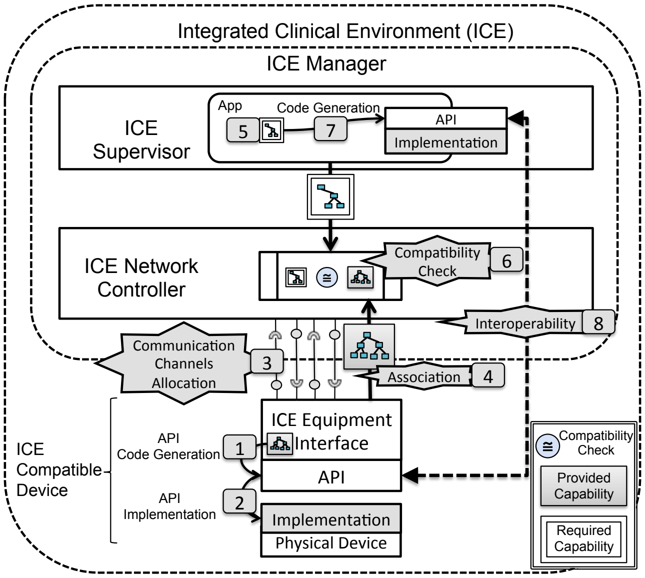

Figure 2: App-Device Interactions on Different Levels of Abstraction

3. PCA SAFETY INTERLOCK EXAMPLE

A Patient-Controlled Analgesia (PCA) pump is a medical device often used in clinical settings to intraveneously infuse pain killers (e.g., opioids) at a programmed rate into a patient's blood stream. A PCA pump also includes a button that can be pushed by the patient to receive additional bolus doses of drug – thus allowing patients to manage their own pain relief. Despite settings on the pump that limit the total amount of drug infused per hour and that impose lock out intervals between each bolus dose, there is still a risk of overdose when using PCA pumps.

ICE can be used to implement an integrated clinical system (ICS) in which an PCA Infusion Monitoring and Safety Interlock App (short: Interlock app) that integrates with monitoring devices to obtain physiological parameters such as blood oxygen saturation (SpO2), End-Tidal carbon dioxide (EtCO2), and respiratory rate (RR) from monitoring devices that are useful for monitoring for respiratory depression (an indication of PCA overdose). The app implements a safety interlock by halting the pump's opioid infusion (via its network interface) when the monitored physiological parameters satisfy a *halt condition*, i.e., have values and/or trends that may signal the onset of a respiratory depression. When a *halt condition* is detected, the Interlock app also provides both a visual and audio alarm through the user interface of *Supervisor*. The ICS is designed for interoperability, e.g., monitoring devices from different manufacturers can be integrated, as long as their network-exposed capabilities are compliant with an ICE DM that satisfies the app's

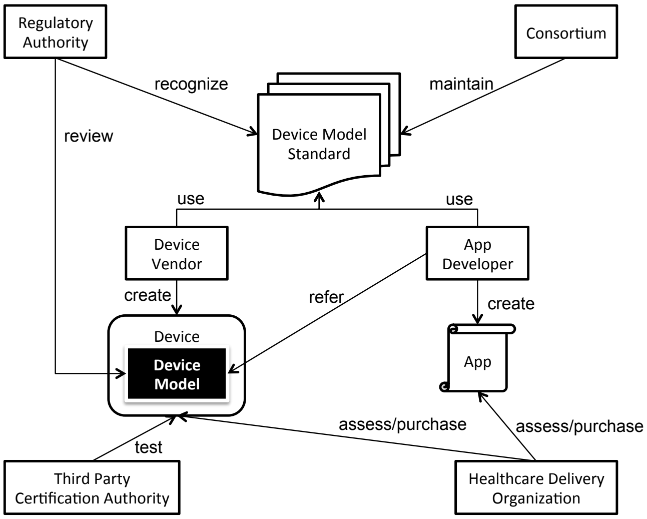

Figure 3: Stakeholders

requirements.

Because the PCA monitoring and safety interlock use case of ICE has been subject of a number of demonstration in the ICE community as well as a subject of current standardization activities, we use it as an example to illustrate aspects of stakeholder activities and the ICE DM.

4. ENVISIONED ICE DM STAKEHOLDERS

Figure 3 depicts the different stakeholders of the device model concept, along with their associations; each stakeholder has certain interests in and usages of the device model, as described below. The stakeholders' needs drive the development of high-level requirements of the DM concept.

Consortium: The envisioned ICE consortium[2] is a group of stakeholders who cooperatively develop standards for the ICE architecture and associated interfaces to support safe interoperability and to facilitate the development of a commodity market of ICE components. The consortium may develop additional artifacts to manage the ICE ecosphere such as development and compliance testing tools, common component libraries, and risk management guidelines.

For the ICE DM, the consortium defines and standardizes a *domain modeling vocabulary*, i.e., "building blocks" consisting of model schemas for common device interface functions, common types for data representation, nomenclature, etc.. Using the domain vocabulary, the consortium will also develop and standardize *reference device models for common classes of devices*, e.g., pulse oximeters, capnography devices. Since common device functions may be realized on different notions of devices (e.g., a SpO2 information may be supplied by a standalone pulse oximeter or a multi-monitor), the consortium will also produce reference models at a finer granularity focusing on capabilities such as common data streams for physiological parameters, alerts, patient data, etc. Extensibility mechanisms within the DM framework will allow vendors extend the reference models with additional features specific to their devices. Consortium standardization on commonalities promotes interoperability, while pursuing finer granularity and extensibility (e.g., to allow device vendors to expose product-differentiating features when necessary). It provides flexibility and avoids "least common denominator" solutions.

Both the domain vocabulary and reference DMs evolve as new device capabilities, safety requirements, and communication technologies are introduced by consortium members, and it is the responsibility of the consortium to manage this evolution. To aid in management, the consortium will organize DM elements with associated ontologies to help maintain clear conceptual undestanding and intent – thus, facilitating the use of the modeling elements by device manufacturers as well as app developers. The consortium will also use *product line engineering* [13] to effectively manage the *commonalities* and *variabilities* of capabilities across the space of device network interfaces.

The consortium would also ensure that supporting infrastructure is developed including official mappings of ICE DM constructs to various programming languages, precise specifications of compliance (e.g., reference models comply with domain vocabulary, vendor extensions comply with references), and precise specification and implementation of app DM to device DM compatibility.

Example: Independently of a particular ICS, the consortium defines the domain vocabulary (i.e.., how physiological data, alerts, etc. should be structured). They would then use this vocabulary to define reference models for specific common physiological parameters (e.g., SpO2, EtCo2, and RR), controls (e.g., PCA enable/disable infusion), alerts (e.g., limit alarm), prescription settings for PCA (e.g., drug name and concentration, loading bolus, patient bolus dose, basal infusion rate, patient bolus lockout interval, hourly dose limit and etc.). Reference models for these finer-grain elements would be used to construct reference models for classes of devices (e.g., PCA pump, pulse oximeter and capnometer). Taking into account an anticipated range of devices and applications, reference models for pulse oximeter and capnometer might define possible communication patterns to access device capabilities, e.g., physiological parameters should either be published periodically or returned upon request (e.g., supporting a "spot-check" function) . Product-line/feature modeling annotations might be used to indicate that the spot-check feature is optional for SpO2 and Pulse Rate in the pulse oximeter reference model, but that periodic publishing is required. The reference models would also support security features, e.g., PCA pump reference model might mandate that PCA prescription setting information is communicated with an option of role based access control description. (e.g., if the setting is writable, it could be changed by clinician).

Device Vendors: The device vendor leverages the ICE DM framework in multiple ways. First, the vendor uses the framework model elements to specify the capabilities of the device, along with associated risk-related attributes and security control features, that are offered over its interface. We refer to the DM for a particular manufacture model of a device (e.g., a Nellcor Oximax N-65) as the *product model.* If a reference model already exists for the class of the device (or for its finer-grained constituents), the vendor uses (and possibly extends) those to form the product model. Second, the vendor uses consortium-provided standardized code generation facilities to translate the product model into programmatic interfaces (APIs) that the vendor will implement

[2] We use "ICE consortium" in this paper to represent an envisioned organization that has yet to be developed. The standards development role for the consortium may take place within existing standards development organizations.

to convert the data/control formats native to the device to ICE standard protocols. Standardized API generation for device/app interfacing facilitates device/app interoperability and compliance of device interface implementations to device models.

Third, the vendor uses consortium-approved testing tools to generate test harnesses customized to their device models, and, in some cases, actual tests (e.g., fuzzing tests) can be auto-generated. This can support the development of assurance artifacts demonstrating device compliance to device models. As the testing tools are shared between the device vendor and the certification authority (described below), many of the compliance demonstration tasks can be pushed out to vendors and incorporated in earlier stages of development and confirmed later by the certification authority. Finally, the device model framework can support construction of regulatory submission artifacts. As in the test suite generation, regulatory-relevant information including risk-related attributes of capabilities could be extracted from the device model and auto-generated to be included in the submission documents for approval. This can reduce submission preparation effort and can facilitate regulatory reviews by presenting ICE interface safety and essential performance properties in an uniform manner across vendors.

Example: Vendors of devices in the Interlock app example utilize the reference models for PCA pump, pulse oximeter, and capnometer. To specify features particular to their devices, vendors configure *variabilities* in reference models (e.g., decline to support the spot-check option on the pulse oximeter, opt out basal infusion rate in PCA prescription setting) and extend their models with extra features such as limit alarms and PCA enable/disable infusion control. After completing these the product models, using ICE DM framework, vendors generate APIs from the product DMs into a programming language supported by the framework, and they derive test cases for compliance checking between the product DMs and device's actual behaviors (e.g., checking the frequency of publishing of physiological parameters, response of PCA prescription setting, and etc.). For the fully implemented devices, vendors prepare documentation for third party certification and regulatory review; ICE DM framework is used to extract risk-related information such as access control information of PCA infusion control and PCA prescription settings and to translate into review-friendly format.

App Developers: The App Developers utilize consortium-standardized reference model elements to specify the device capabilities that apps require to perform their clinical function. We refer to this specification, phrased in terms of the ICE DM framework as the *app ICE DM requirements* (which we will safely shorten to "app requirements" and ignore the obvious semantic conflict with the notion of "app's full functional and safety requirements" since the latter are outside the scope of this paper). As with Device Vendors, App Developers use consortium-provided tooling to: (a) auto-generate APIs for interfacing with devices that are compliant with app requirements, (b) support testing and verification of app interactions with devices, and (c) support construction of artifacts necessary for safety reviews.

Example: In the Interlock app case, in addition to implementing the basic features of the app such as halting condition logic and displaying various physiological information and alerts, the App Developer specifies the app requirements on device services needed to support the app's clinical function, e.g., PCA pump with an infusion enable/disable control, and data streams for SpO2, EtCo2, and RR with limit alarms. Also, the developer specifies required quality of service constraints (e.g., rate at which physiological parameters will be published). Using the consortium code generation tools, the app requirements are translated into APIs that are used to access the specified required device services. Similar to device vendors, testing, documentation, and regulatory submission tasks are supported using the ICE DM framework.

Third Party Certification Authority: The Third Party Certification Authority certifies that ICE component implementations are compliant with both ICE implementation and safety/security standards. For devices, this includes certifying that the behavior of the medical device as exposed over its network interface is compliant with the device's declared DM product model. Evaluation of compliance is performed using compliance testing tools approved by the Consortium (perhaps extended with additional testing developed by the Vendor or Third Party Certifier).

Example: The certifier would evaluate submissions from vendors supplying the PCA Pump as well as devices supplying SpO2, EtCO2, and RR physiological parameters. They would use consortium-approved testing tools to confirm tests previously performed by vendors and would use testing and quality assurance methods to determine if the devices were compliant with their (a) declared DM product models and other ICE implementation standards and (b) safety/security standards relevant to ICE. As part of the evaluation, the certifier assesses risk management artifacts provided by the vendor. The same process would be applied to the app as well. A successful evaluation would result in a digital certificate for the vendor to use that attests to the certification, and would produce documentation that would be included in a subsequent regulatory submission for the devices and the apps.

Regulatory Authority: The Regulatory Authority regulates the safety and effectiveness of systems built using the ICE platform. The ICE community's goals include having the Regulatory Authority officially recognize the standards relevant to ICE including the standardization of ICE DM. The Regulatory Authority would also need to recognize the competency of Third Party Certifiers to evaluate compliance of devices to the recognized ICE standards (or in some cases, may accredit the Third Party to perform safety reviews).

A part of the Regulatory Authority's review activities for a device would be to review safety of the ICE network interface of the device as specified by the device DM product model. The review relies on documentation/evidence produced during third-party certification. As part of the safety review, the reviewer would be interested in identifying the risk related parameters exposed on the device's network interface and their association with particular hazards. The ICE DM framework enables Vendors to include machine-readable annotations in a DM product model to provide risk-related information for individual parameters. ICE DM framework report generation can facilitate production of information for supporting compliance to risk management standards such as ISO14971 or IEC80001.

Example: As part of the conventional review process for stand-alone devices include the PCA Pump, Pulse Oximeter, and Capnometer that support the PCA Interlock ICS, the Regulatory Authority would review ICE DM product models for these devices, the safety/security-related aspects of these models, along with evidence of compliance of the implementation to the models.

Health-care Delivery Organizations: A health-care delivery organization (HDO) purchases devices and connects them with the ICE platform. Rather than having the HDO staff rely on product technical descriptions to judge the suitability of connection with the platform or a particular app, the HDO can rely of the precise characterization of device capabilities provided by the vendors' ICE DM product models. Previous Third-Party Certification enables high levels of trust that device implementations with those specifications. The Consortium-specified and third-party-tested implementation of the platform capability check between DM-specified app requirements and device capabilities provides high levels of confidence in app/device interoperability and supports confidence in the overall safety of the integrated system. Note that the platform's use of the ICE DM product model to support compatibility checking and configuration of platform services is transparent to the HDO staff assembling the ICE system.

The device model framework could also be used to support device procurement. In particular, hospital may assess or place requirements on devices to be purchased by way of the capabilities of devices expressed through their device models. The current practice focuses on manual assessment of a device's capabilities from marketing materials and technical specifications. However, if the device models of all the devices are available in a database, support for assessment of devices matching the HDO's needs could be significantly automated.

Example: When the Interlock app is purchased by HDO, ICE DM could be utilized in procuring compatible devices (or vice versa – HDOs could assess whether they can use apps on the market based on a comparison with product models of pump, pulse oximeter, and capnography devices that they have in their inventory).

Assuming a clinical case occurred in an HDO which is equipped with the Interlock app and devices from the example, the clinician prepares a PCA pump and monitoring devices and connects them to the ICE with authorization. When devices are connecting to ICE, ICE product DMs are transmitted and stored to the *Network Controller*. The clinician configures the Interlock app with the patient profile through a user interface provided by *Supervisor* and sets the *halting condition*. At the launch of the app, *Supervisor* extracts DM-based app requirement and sends a request for compatible device resources to the *Network Controller*. *Network Controller* searches through the connected devices and builds a list of devices that are compatible to the app requirement using compatibility checking algorithm. The basic principle of compatibility checking is to find devices with a superset of capabilities than what is required. In this example, a pulse oximeter that publishes SpO2 without a limit alarm is excluded where a pulse oximeter that publishes Pulse Rate as well as SpO2 with a limit alarm will be listed as a compatible device. As *Network Controller* returns the list of compatible devices per items of app requirement, the clinician selects the device combination to start the operation of the app. Label 6 in Figure 2 illustrates the compatibility checking process and dashed arrow with label 8 shows the usage of implemented APIs for communication between an app and devices during an app operation.

5. ICE DM REQUIREMENTS

From the use of stakeholders, ICE DM should be capable of expressing information involved in supporting app functionality, app-device compatibility, automatic checking of manager-device compatibility, and assessing safety. Table 5 lists the requirement categories.

Data: The success of a clinical system composed of communicating devices and apps hinges on *the ability of a device/app to produce data in a form that captures the necessary information and can be consumed and unambiguously interpreted by other devices/apps in the system.* For example, a capnometer measuring EtCO2 data every second in mmHg should be able to produce data in a form that a monitoring application can consume and interpret as EtCO2 measured in mmHg.

To enable such seamless data representation, *ICE DM should support well-defined description of rich and structured data required to represent the variety of data used in clinical environment.* Besides the basic support for describing scalars and value sequences, it should also support the description of structured data (akin to records in most programming languages).

Most often, software in devices/apps are composed in various programming languages depending on the vendors of the devices/apps. To allow the communication between such heterogeneous components, the *support for data description should be programming language agnostic and provide well-defined translation schemes for various programming languages* (to prohibit misinterpretation of data).

Since there are (and will be) numerous data types specific to medical/clinical domain that will be reused by both app and device vendors, *ICE DM should provide first class support for such domain specific data types* to reduce redundancy and ensure consistency in ICE ecosphere. Few examples of such data sorts are listed in the table below.

Data Sort	Description
Device Properties	Collection of attributes that describes, e.g., the manufacturer, the device model number, and the device type within Consortium-defined nomenclature; this may proceed to fine levels of granularity including specific sensors/actuators on the device.
Physiological	Data gathered by the sensing capabilities of devices including simple or a compound numeric value, waveforms, image data, etc.
Setting	Parameters determining the operation/control of the device (e.g., alarm ranges, infusion rate for PCA pump).
Alarm	Alarm events that notify the caregiver of exceptional events or warnings.
Control-Operations	Control information to implement a medical application involving closed-loop control, or safety interlock.

To enable device vendors to differentiate their offerings, *ICE DM should allow extension of existing data descriptions in an interoperable way*, e.g., a pulse oximeter from vendor X may report SpO2 data along with its deviation from the previously reported SpO2 data.

DA1: ICE DM should support well-defined and machine-readable description (e.g., specified using an unambiguous formal grammar) of rich and structured data required to represent the variety of data used in clinical environments.

Category	Requirement
Data	Flexible/Extensible description of structured data in a well-defined language that enables automation (via formal grammar) and ensures unambiguous data representation.
Communication	High-level description of offered and required communication (via a set of communication patterns).
Quality of Service	Unambiguous, composable, and device/app local description of QoS properties of aspects such as communication and performance.
Security	Unambiguous, composable, and fine-grained description of access control to different parts of a device interface in different device states (via RBAC).
Safety	Unambiguous description of safe and unsafe states of a device along with how the device interface specification changes in these states.
Behavior	Unambiguous description of states and transition between states of a device in terms of interface elements (via pre/post conditions, invariants, ordering constraints, etc.)
Modeling Process and Environment*	Reuse of models (via library of DM prototypes, partial instantiation), support traceable iterative refinement of models, support automatic generation of auxiliary artifacts (such as documentation and tests) and support conformance checking (via matching the features of two device models).

Table 1: Requirement Categories

DA2: ICE DM should support programming language-agnostic data description along with (a) well-defined translation schemes into common programming languages (with associated code generation for marshaling and un-marshaling data) used in developing ICE apps and ICE devices interfaces, as well as (b) report auto-generation that facilitates reviewing of DM capabilities and meta data.

DA3: ICE DM should provide first class support for data types specific to medical/clinical domain. Also, the Consortium should play a role in standardizing the data "nomenclature". The data attributes and nomenclature should include definition of units and medical interpretations of data (e.g., annotation about units, usage examples).

DA4: ICE DM should allow extension of existing data descriptions (e.g., deriving new types by adding additional fields to previously defined types).

Communication: In a typical instance of ICE, devices and apps that communicate to enable specific clinical scenarios, e.g., coordinate the operation of ventilator and x-ray machine to ensure the ventilator is turned off only when the x-ray is being captured. Such communication stems from the *need to exchange data and control commands between devices and apps.*

To enable interoperability of heterogeneous devices and apps, *ICE DM should support the description of communication needs of devices while abstracting away low-level details of realizing communication.* Specifically, a device model should be able to capture the key aspects of communication relevant to clinical systems, e.g., lifetime and frequency of data, data stream identifier. Further, it should support iterative refinement (or automatic translation) of these aspects into implementation details (e.g., derive tolerable network latency from data frequency and lifetime, derive URI from data stream identifiers) that are necessary for devices to connect to an instance of ICE and communicate with other devices and apps in the environment.

Most often, communication in a domain can be distilled down to few common recurrent patterns. In the domain of clinical system, distribution of data, subscription to data, request of data, request of action, and setting of parameters are common recurrent communication needs. So, as means of abstraction, *ICE DM should support a set of communication patterns that abstracts common communication needs in clinical systems* such as: (i) periodic/aperiodic publishing of information from device, (ii) app request of data from a device, (iii) app commands to change settings on a device, (iv) app commands to invoke an operation on device with an optional return value, and (v) higher-level notions of transaction that ensure consistent state of exchange across multiple devices.

With well-defined semantics, these patterns can ease the burden of reasoning about ICE-based systems. Further, these patterns can help decouple the device model from the capabilities of communication substrates (or middleware).

CO1: ICE DM should support description of communication needs of devices while abstracting away low-level details of realizing communication.

CO2: ICE DM should support a set of communication patterns that abstract common communication needs in clinical systems

CO3: ICE DM should be mappable to commonly used communication substrates (e.g., DDS, various event buses).

Quality of Service: A clinical system is effective only when it provides service in a timely manner, e.g., alert the attending nurse about the drop in a patient's heart rate, deliver a dose of analgesic when the patient presses the button. Such timely service is only possible if *devices and apps state and fulfill the quality of offered/used services and the combination of these quality guarantees is sufficient to guarantee the quality of service required of the clinical system.*

To reliably fulfill such guarantees in composite clinical systems, *ICE DM should support description of the quality of both offered and used services.* For example, a description of a pulse oximeter should include information about the maximum latency to service a request for pulse rate. Note that some aspects of communication can be deemed as quality of service guarantees. Few examples of QoS properties are given in the table below.

Property	Description
Minimum Separation	Minimum duration between two consecutive communication events (e.g., data requests).
Maximum Latency	Maximum duration to service a (data/action) request.
Minimum Remaining Lifetime	Minimum duration for which the data should be valid.

Since clinical systems in ICE are dynamic composites of devices and apps, *QoS properties captured in a device model should be composable with QoS properties captured in other device models.* Further, QoS properties should be *local to devices* to allow flexibility while composing clinical systems during deployment. Further, to ensure interoperability, *QoS properties should be unambiguous*, e.g., using a standardized set of QoS properties and units. For example, instead of allowing the specification of the rate of data publication both in terms of publication frequency and time interval between consecutive publications, standardize the rate of publication to be specified in terms of the interval between consecutive publications expressed in milliseconds.

QU1: ICE DM should support description of the quality of both offered and used services.

QU2: QoS properties specified device models should support evaluation of app/device QoS compatibility as well as determination of end-to-end (system) quality of service.

QU3: QoS properties should be unambiguous.

Security: In a HDO, employees have different privileges regarding their ability to access data, to provide treatment of patients, and to author orders for care-giving. For example, while a clinician may be authorized to monitor a patient receiving PCA infusion, but he/she is not allowed to modify prescriptions or to program the PCA pump. Further, some apps, e.g., those that support remote monitoring or mobile notification to clinicians may have data privacy requirements (via HIPAA) that restrict access to certain forms of data. While ICE DM is publicly visible to all stakeholders, app access to device operations and data should be configurable. To handle these varying privileges and privacy restrictions, *ICE DM should support description of access control to different parts of different interfaces of a device in different device states, e.g., via role based access control (RBAC).*

In a similar vein, when components of clinical systems communicate via a network, they are susceptible to security attacks such as attempts to steal sensitive information or take control of the system or its components. To protect against attacks, *ICE DM should support description of authentication and encryption constraints on communication.*

SE1: ICE DM should provide role-based access control mechanisms to constrain access of apps to device data and operations. This includes the ability to configure read/write access to data fields (and groups of fields).

SE2: ICE DM should support description of authentication and encryption constraints on communication.

Safety: An ICE-based ICS utitlizes medical devices that can also function as stand-alone entities – each with its own specific set of safety goals. The ICE app that defines the ICS depends on the safety and reliability of the device components, as well as the reliability of the platform components, to achieve the overall safety goals of the ICS. Moreover, since ICE aims to support a compositional approach to safety for plug-and-play interoperability, ideally, one should be able assure the safety of the ICS relying only on properties which are exposed on the interfaces of components (as well as the knowledge that component interfaces comply with component interface properties). This means that various safety-related aspects should be captured on device interfaces. First, the device's interface-exposed functions need to be captured in the ICE DM product model at a sufficient level of precision (e.g., using the behavioral specification discussed in 'Behavior' section as to support system functional and safety assurance. Second, if there are faults/failures of the system context in which the device is used (e.g., an accidental disconnection of the device from the network controller), the app needs to be able to move the device to a state that is safe for its standalone function and use within the current ICS. Upon loss of connection from the network controller, the device similarly needs to be able to move itself to a safe state (where the safe state may be pre-configured by the app in accordance with current use of the device). For example, a device response to a network failure may be *fail operational*, *fail passive*, *fail secure*, *fail safe*, etc.. Finally, the device itself may fail, and the risk management process for the ICS (driven by the app developer) should know the possible ways that device failure may give rise to errors and faults propagating from the device into the ICS via the device's network interface. This and the previous item of information are necessary to support envisioned compositional approaches to hazard analysis.

Category	Description
Data	Numerical or (in)equality constraints on parameters or returned values.
Event Notification	Constraints on a component's state when it publishes an event.
Component Invariant	Constraints on a component/system's state in every non-intermediate state of the component/system.
State/Mode*	As component behavior often depends on state or mode, the current value of a component's state/mode is exposable on its interface.
Pre/Post-condition	Constraints on inputs and system state before invoking an operation and on outputs and system state after completing the operation.
Temporal Ordering	Constraints on the ordering of operations on an interface.
Timing Properties	Constraints on the Real-time aspects of system actions; constraints on how variable values compare to those in previous periods.

Table 2: Behavioral Interface Specifications

SA1: ICE DM should provide the ability to declare information indicating the "safe state" that a device will transition to if network connectivity is lost. The ICE DM may also allow an app that gains exclusive device access to select one from among several possible safe states.

SA2: ICE DM should provide annotations for device vendors to designate risk-related DM data fields/operations along with criticality of those fields/operations.

SA3: ICE DM should provide annotations for vendors to characterize how data and communication from a device may be affected by device failure modes, as well as other reliability information.

Behavior: In the end of Section 2, we explained how the ICE goal of plug-and-play interoperability in the context of compositional construction and safety reviews necessitates the inclusion of behavioral information on component interfaces. In our opinion, the only way to achieve high levels of assurance in such reviews is by capturing behavioral information in terms of a behavioral interface specification language (BISL) [5] based on *formal*, *logic-based* specifications. Table 2 gives examples of the types of properties that the BISL should be able to capture. When a device vendor develops a DM product model, important behavioral aspects of the device's function would be captured in the BISL. Similarly, an app vendor will annotate their DM-based device requirements with behavioral specifications via the BISL. When an app is integrated with its required devices to form an ICS, the device's behavioral specification should be compliant with the app's requirements. Enforcement of behavioral compliance can be achieved (a) a priori, before integration (but this limits an app to work with a specific set of device interfaces), (b) via automatic analyses during association, e.g., in the MDCF, during the app DM/device DM compatibility check, or (c) during run-time through dynamic monitoring (but this has the disadvantage of needing to halt the app's function if violations occur).

BE1: ICE DM framework should include a formal Behavioral Interface Specification Language that can capture properties such as those in Table 2.

BE2: ICE DM BISL should include a framework that can guarantee the compliance of device behavioral specifications to an app's specification of behavioral requirements.

Modeling Process and Environment: Previous sections have described the types of information that should be captured in a device model and how that information might be represented. In this section, we turn to broader issues such as designing the framework to encourage industry adoption, to facilitate management of framework/consortium assets, and to ensure the correctness and integrity of device models as they are used by different stakeholders.

Machine-readability of device model is a fundamental property that underpins many other DM requirements. Machine-readability enables automation, for example, to provide tool-support for capturing device models and app requirements, automatically assessing compatibility of app requirements and device capabilities, checking conformance of product models to consortium-defined reference models, generation of APIs from device models, providing support for testing automation, and auto-generation of artifact templates supporting certification and regulatory submissions. At the same time, *device models must be human-readable/writable – to the point of being easy to understand and easy to author by stakeholders.* This is necessary, e.g., to use the device model as the basis for assessing capabilities to guide acquisitions (e.g., for HDOs) for evaluating safety/security and implementation compliance (e.g., Third-Party Certifiers, and Regulatory Authorities). Existing modeling and domain-specific language approaches provide a good basis for bridging the gap between the need for a formal high-level easy-to-comprehend view and an efficient machine representation that stored on devices and transferred across networks.

Extensive experience in the technology field indicates that *providing quality tool support for a new language that addresses development/authoring, testing/verification, and that provides multiple ways for stakeholders to leverage the language within their workflows significantly improves adoption of the language.* For this reason, the consortium should be careful to invest strategically in the development of such tools. Other industry experience indicates that *providing a platform with a broad collection of assets (e.g., building blocks for building a variety of applications) also encourages adoption.* Thus, the consortium needs to provide a variety of commonly-used reference model elements to facilitate construction of feature product models of vendor devices. When providing reference models, the consortium needs to standardize features that are *common* across a broad class of devices while providing a means for vendors to specify product-differentiating *variabilities.* 11073 provides mechanisms for declaring optional features and for adding additional features, but those mechanisms suffer from being too simplistic (e.g., cannot formally aggregate optional elements into named feature sets, cannot specify the constraint to select one option from a list of available options) and cannot formally capture dependencies or anti-dependencies between optionality choices (e.g., if one feature is present another must also be present; if one feature is present, another must be absent). Significant progress in product-line and feature modeling [13] can be immediately applied to address these needs and overcome the limitations of 11073. Overall, the consortium needs to embrace the fact that they are helping to facilitate the development of a cross-vendor *product line of device interfaces* and intentionally evaluate the use of product-line engineering concepts to manage the DM framework.

Finally, *the entire DM framework should be carefully designed to support a variety notions of machine-checkable compatibility and compliance.* As noted in earlier sections, assurance cases for ICE-based integrated clinical systems rely on the fact that, when integrating components to support a clinical application, the assumptions that an ICE app makes about a component's capabilities must be determined to hold true. It is our view that for mid- to high-criticality applications, the integration assurance must be established by highly trusted algorithms that can automatically determine if formally-specified integration constraints are satisfied – because this is the only way that we know of for establishing sufficient levels of trust in the correct integration of safety-critical dynamic plug-and-play interoperable systems. Difficulties in achieving this goal can be remedied by reducing the degree of "dynamicity" of the plug-and-play aspect or by reducing the variability in interfaces (e.g., allowing integration verification obligations to be discharged using more conventional means since the standardized features will be known at component development time).

In addition, *the DM framework should be designed in a multi-tiered structure to cleanly and explicitly support the natural phasing of the construction of device models across stakeholders.* Specifically, we described in Section 4 how the consortium begins with a domain-independent modeling framework [Tier 0] and uses that framework to define a *modeling vocabulary* [Tier 1] for the domain of medical device interfaces. The consortium uses the modeling vocabulary to construct *reference models* [Tier 2]. Vendors use the reference models to construct *product models* [Tier 3] and then *instance models* [Tier 4] for individual devices. Between each of these tiers (see Figure 4 there is a notion of *compliance*, and we argue that each compliance notion should be machine verifiable. The modeling vocabulary must satisfy the syntax and structure constraints of the modeling framework [Tier 1 to Tier 0 compliance]. References models be well-formed from elements of the modeling vocabulary [Tier 2 to Tier 1 compliance]. While these two notions are achieved by the consortium (i.e., highly-skilled individuals with significant experience with the modeling framework), the construction of product models is carried out by individual vendors with varying skills and experience. Therefore, it is especially critical that the compliance of product models to reference models [Tier 3 to Tier 2] and instance models to product models [Tier 4 to Tier 3] be machine checkable. 11073 PHD (Personal Health Device) describes a compliance framework (e.g., when vendors are developing instance configurations of device specializations), but the notion of compliance is manually documented and evaluated. Even

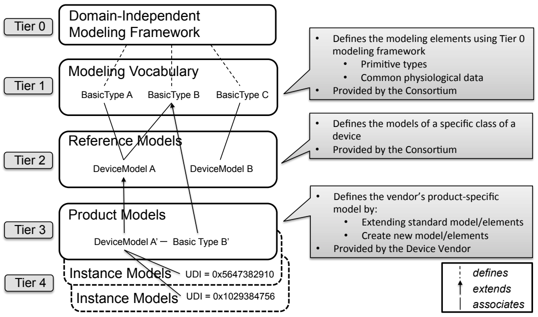
Tier 0
Domain-Independent Modeling Framework
Tier 1
Modeling Vocabulary
BasicType A
BasicType B
BasicType C
Defines the modeling elements using Tier 0 modeling framework
Primitive types
Common physiological data
Provided by the Consortium
Tier 2
Reference Models
DeviceModel A
DeviceModel B
Defines the models of a specific class of a device
Provided by the Consortium
Tier 3
Product Models
DeviceModel A'
Basic Type B'
Defines the vendor's product-specific model by:
Extending standard model/elements
Create new model/elements
Provided by the Device Vendor
Tier 4
Instance Models
UDI = 0x5647382910
Instance Models
UDI = 0x1029384756
defines
extends
associates

Figure 4: Multi-level Refinement

though organizations like NIST are developing validators for machine-readable XML-based schemas of domain information models, the compliance checking abilities between conceptual tiers (which are not cleanly called out in 11073) are quite limited, e.g., due to the informal aspects of 11073 schemas for optionality, inter-attribute dependencies, etc.

MO1: Device models and app requirements should be machine-readable.

MO2: ICE DM framework should be readily accessible and understandable by domain experts/stakeholders.

MO3: ICE DM framework should provide authoring tool for ICE DM that will assist syntax highlighting, code completion, and validation.

MO4: ICE DM framework should provide transformation tools that convert device models to various artifacts such as executable API, regulatory review documentations, and test cases. Also, the tools should be capable of trace back from the artifacts to the source device models.

MO5: ICE DM framework should provide well-organized collection of DM reference models.

MO6: ICE DM framework should support capturing commonality and variability in device capabilities and app requirements, which aligns with the concept of the feature modeling in the software product-line engineering [13].

MO7: The definition of compatibility between two ICE DMs should be formally defined by *Consortium* and supported by a reference implementation of a compatibility checking algorithm.

MO8: The consortium should specify the notion of compliance between an implementation of a network interface and a DM product model and should establish testing and verification procedures for demonstrating compliance.

MO9: ICE DM should support explicit modeling tiers aligned with stakeholder activities and with automatic compliance checking between tiers.

6. CONCLUSION AND FUTURE WORK

We have proposed high level requirements for the ICE DM framework based on the needs of ICE stakeholders. Some aspects of our work have been inspired by a detailed analysis of 11073. However, majority of requirements were elicited through understanding ICE stakeholder needs, especially, the needs related to supporting compositionality, safety, security, and more advanced type systems which are deficient in 11073.

As an immediate next step, we are prototyping a modeling tool for specifying device models as described in this paper. In addition, we are conducting a gap analysis for 11073.

Beyond the above steps, technologies to implement each of the proposed requirement category needs to be identified and evaluated, e.g., BLESS [12] as the language to specify behavior. In addition, while we have validated the above requirements in MDCF by prototyping ICE DM and describing interfaces of various devices phrased in terms of the DM, a broader class of devices needs to be explored to vet and reinforce the proposed requirements.

7. REFERENCES

[1] Health Level Seven. https://www.hl7.org/implement/standards/.

[2] IHE Patient Care Devices. http://www.ihe.net/Patient_Care_Devices/.

[3] ASTM International. ASTM F2761 - Medical Devices and Medical Systems - Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE), 2009.

[4] J. Hatcliff, A. King, I. Lee, A. MacDonald, A. Fernando, M. Robkin, E. Vasserman, S. Weininger, and J. M. Goldman. Rationale and architecture principles for medical application platforms. In *Cyber-Physical Systems (ICCPS), 2012 IEEE/ACM Third International Conference on*, pages 3–12. IEEE, 2012.

[5] J. Hatcliff, G. T. Leavens, K. R. M. Leino, P. Müller, and M. Parkinson. Behavioral interface specification languages. *ACM Comput. Surv.*, 44(3), June 2012.

[6] ISO/IEEE. *ISO/IEEE11073-10101:Nomenclature*. 2004.

[7] ISO/IEEE. *ISO/IEEE11073-10201 Health informatics - Point-of-care medical device communication*. 2004.

[8] ISO/IEEE. *ISO/IEEE11073-20601:Application Profile - Optimized exchange protocol*. 2010.

[9] A. King, D. Arney, I. Lee, O. Sokolsky, J. Hatcliff, and S. Procter. Prototyping closed loop physiologic control with the medical device coordination framework. In *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care*, pages 1–11. ACM, 2010.

[10] A. King, S. Chen, and I. Lee. The middleware assurance substrate: Enabling strong real-time guarantees in open systems with openflow. In *Object/component/service-oriented realtime distributed computing (ISORC), 17th IEEE Computer Society symposium on*. IEEE, 2014.

[11] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, and S. Weininger. An open test bed for medical device integration and coordination. In *Proceedings of the 31st International Conference on Software Engineering*, 2009.

[12] B. Larson, P. Chalin, and J. Hatcliff. BLESS: Formal specification and verification of behaviors for embedded systems with software. In *NASA Formal Methods Conference, 2013. Proceedings of*, pages 276–290. NASA, 2013.

[13] K. Pohl, G. Böckle, and F. J. v. d. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[14] J. M. Rushby. Design and Verification of Secure Systems. *Operating Systems Review*, 15:12–21, 1981.