EXECUTABLE CLINICAL MODELS FOR ACUTE CARE

BY

MARYAM RAHMANIHERS

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Abstract

Medical errors are the third leading cause of death in the U.S., after heart disease and cancer, causing at least 250,000 deaths every year [1]. These errors are often caused by slips and lapses, which include, but are not limited to delayed diagnosis, delayed or ineffective therapeutic interventions, and unintended deviation from the best practice guidelines. These situations may occur more often in acute care settings, where the staff are overloaded, under stress, and must make quick decisions based on the best available evidence. An *integrated clinical guidance system* can reduce such medical errors by helping medical staff track and assess patient state more accurately and adapt the care plan according to the best practice guidelines.

However, a main prerequisite for developing a guideline system is to create computer interpretable representations of the clinical knowledge. The main focus of this thesis is to **develop executable clinical models for acute care**. We propose an organ-centric pathophysiology-based modeling paradigm, in which we translate the medical text into executable interactive disease and organ state machines. We formally verify the correctness and safety of the developed models. Afterward, we integrate the models into a best practice guidance system. We study the *cardiac arrest* and *sepsis* case studies to demonstrate the applicability of proposed modeling paradigm. We validate the clinical correctness and usefulness of our model-driven cardiac arrest guidance system in an ACLS training class. We have also conducted a preliminary clinical simulation of our model-driven sepsis screening system.

Acknowledgments

This project would not have been possible without the support of many people. I would like to sincerely thank my PhD advisor Prof. Lui Sha for guiding and supporting me over the years. It is hard to find words to describe how much I appreciate all he has done for me as an advisor and a mentor. I would like to thank my thesis committee members, Dr Sandy Weininger, Prof. Rahul Mangharam, Prof. Alex Kirlik, Prof. Carl Gunter and Prof. Tarek, Abdelzaher for all of their guidance through this process; your ideas and feedback have been absolutely invaluable. I am also grateful to Dr. Richard Berlin and Troy Hoshauer from Carle Foundation Hospital for all their help and support in this work. I also would like to thank my fellow graduate students and collaborators Dr. Mu Sun, Dr. Gheolgi Kim, Dr. Min Young Nam, Dr. Jeonghwan Choi, Dr. PoLiang Wu and Andrew Ou. I am very grateful to all of you.

Finally, I would like to thank and dedicate this thesis to my family. To my father, *Rahim*, I thank you for always being proud of me. To my angel mother, *Parvin*, I am greatly indebted to you for being there for me through these years and supporting me in my determination to find and realize my potential. I thank you for all your prayers; all that I am I owe it to you. To my smart, beautiful, funny and amazing sister, *Mana*, I thank you for always being my best friend and favorite distraction. And lastly to my husband, *Arash*, you are my dearest friend, my biggest comfort, my greatest support, my truest smile and my deepest love. I undoubtedly could not have done this without you.

Table of Contents

Chapter 1

Introduction

According to ”To Err is Human,” a report by the Institute of Medicine (IoM), medical errors are the eighth leading cause of death among Americans causing close to 100,000 deaths every year [2]. According to the most recent study that comes nearly two decades after this IoM report, medical errors has become the third leading cause of death in the U.S., after heart disease and cancer, and annual number of deaths has increased to 250,000 [1]. These *preventable* errors are often caused by slips and lapses, which include, but are not limited to delayed diagnosis, delayed or ineffective therapeutic interventions, and unintended deviation from the best practice guidelines. These situations may occur more often in acute care settings, where the staff are overloaded, under stress, and must make quick decisions based on the best available evidence.

One of the main contributing factors to slips and lapses in clinical environments is the lack of integrated and comprehensive information. According to a study from a Canadian group performed in 2008, the care of critically ill patients generates a median of 1348 individual data points/day and that this quantity has increased 26% over five years [3]. Medical staff must mentally correlate the individual data points they observe from standalone devices. This is an error-prone process due to the limitations of human memory and realtime processing capability. Another contributing factor may be inadequate alarm configuration policies and practices. As reported by ECRI, which is a non-profit health agency, *alarm hazards* are the first item in the list of top 10 health technology hazards that put patients in hospitals at risk [4].

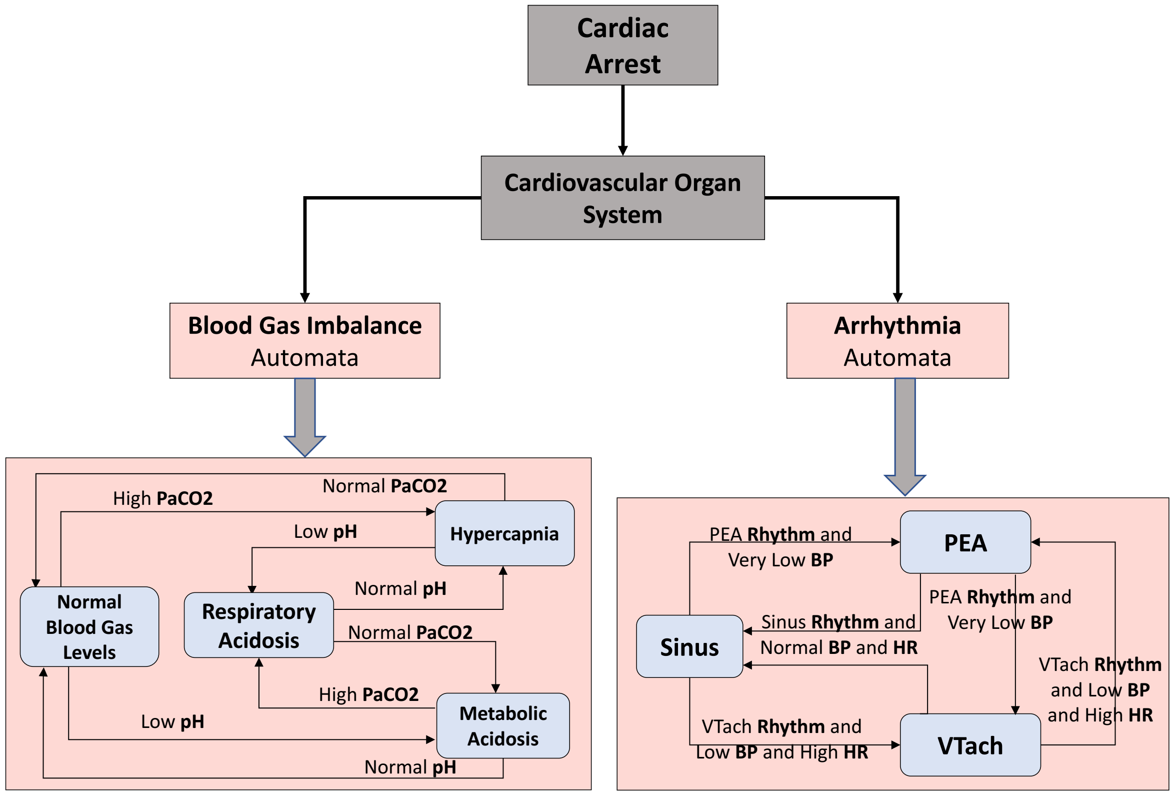
An *integrated clinical guidance system* can reduce such medical errors by helping medical staff track and assess patient state more accurately and adapt the care plan according to the best practice guidelines. However, a main prerequisite for developing a guideline system is creating computer interpretable representations of the clinical knowledge contained in best practice guidelines. In other words, creation of such systems requires considerable amount of **modeling** activity such as deciding what patient data is relevant and identifying the concepts and relationships among them that relate to guideline generations [5]. The main focus of this thesis is on **developing executable clinical models for acute care**.

Such clinical models have been and continue to be studied extensively in medical community. However, the common format used to represent and communicate clinical knowledge, including such models, is English (*text and graphics*). Therefore, utilizing these models when providing clinical care can be as challenging as using a map while driving. Turning maps to GPS based navigation tools has revolutionized driving. Translating pathophysiological models, based in medical best practice text, into executable models can have a similar effect on the use of computer guidance systems in medicine.

We propose an organ-centric pathophysiology-based modeling paradigm, where we translate the medical text into executable interactive disease and organ state machines. This is possible because disease and organ states are already discretized by the medical community, for example, different stages of cancer, diabetes, heart diseases, etc. We model the patient state and different disease stages according to pathophysiology of organ systems, which describes how each of the organ functions are affected by the disease. Furthermore, the diagnosis and treatment are organized for each stage. The developed models can then be integrated in a guidance system and derive its execution. Next, we look at a simple illustrative example before discussing the details of our modeling paradigm.

1.1 Illustrative Example

Cardiac arrest is initially the devastating cessation of cardiac activity, which is caused by the loss of heart's electrical and muscular pumping function. The cardiovascular organ system is the main organ affected by this disease. We model the disease according to the pathophysiology of organ systems, which describes how each of the organ functions are affected by the disease. In cardiac arrest, the



Cardiac Arrest
Cardiovascular Organ System
Blood Gas Imbalance Automata
Arrhythmia Automata
Normal Blood Gas Levels
Hypercapnia
Respiratory Acidosis
Metabolic Acidosis
High PaCO2
Normal PaCO2
Low pH
Normal pH
Normal PaCO2
High PaCO2
Low pH
Normal pH
PEA
Sinus
VTach
PEA Rhythm and Very Low BP
PEA Rhythm and Very Low BP
Sinus Rhythm and Normal BP and HR
VTach Rhythm and Low BP and High HR
VTach Rhythm and Low BP and High HR

Figure 1.1: Organ-Centric Pathophysiology-Based Model of Cardiac Arrest

heart rhythm may change and include abnormal or irregular heart rhythms or waveforms (arrhythmias). Furthermore, insufficient blood circulation in the lungs impedes the CO_2 and O_2 exchange resulting in CO_2 accumulation in the blood, which may be reflected by elevated blood acidity level indicated by decrease in blood pH level. The abnormal pH is revealed in arterial blood gas (ABG) test. These are two examples of pathophysiological changes in cardiovascular organ system caused by cardiac arrest.

To model cardiac arrest, we encode the *arrhythmia* and *blood gas imbalance* processes among others, using finite state machine formalism. In each of these state machines we codify different stages/types of heart rhythm irregularity or imbalance in arterial blood gas levels according to existing medical knowledge. For example, *pulseless electrical activity*, a heart rhythm observed on the electrocardiogram that should be producing a pulse, but is not, is a state in the *arrhythmia* state machine. Similarly, *severe metabolic acidosis*, a very low blood pH, is a state in *blood gas imbalance* state machine. When new blood pressure, heart rate, and EKG readings are received and a blood pH value from a recent ABG test is entered by the nurse, the *arrhythmia* and *blood gas imbalance* state machines will transition to the appropriate states. Let us assume the current readings indicate a very low blood pressure and blood pH, no meaningful heart rate reading and an EKG rhythm that is not a flat line but does not produce a pulse. Therefore, the *arrhythmia* machine transitions to *pulseless electrical activity* state and *blood gas imbalance* transitions to *severe metabolic acidosis*. Figure 1.1 shows the state machines modeling these two pathophysiological processes. Note that we only present a simplified version of cardiac arrest model to illustrate the concept; more details of the proposed modeling paradigm is presented in Chapter 2 and the more elaborate cardiac arrest model is presented in Chapter 3. In the next section, we discuss the technical challenges we have identified for developing executable clinical models for acute care.

1.2 Technical Challenges

As mentioned above, the primary contribution of this thesis is development of executable models of a class of acute care best practice algorithms and the supporting disease models. Therefore, we first present the major *modeling challenges* and our proposed solution(s) to address each.

1.2.1 Modeling Challenges

Challenge 1.a: Medical knowledge representation must be in a machine-processable form. The medical knowledge must translate from text format to a structure appropriate for the logical application of that knowledge by a computer guidance system. The developed clinical model(s) should be executable or must provide a means for automatic execution to facilitate the design of best practice guidance systems.

Proposed Solution: We translate medical knowledge to executable models using an organ-based pathophysiological paradigm. We use finite state machine formalism to develop the models. The models are executable and can directly be used to derive the execution of a best practice guidance system. The details of model semantics is presented in Section 2.2.

Challenge 1.b: The developed models must encode the relevant clinical information. The model needs to encode the medical information or knowledge relevant to the disease under study. For example, a physician often is interested not only diagnosis itself but also the process of diagnosis and the clinical data required to make a diagnosis. However, the encoded information must be adequate but not excessive such that it creates a mental overload.

Proposed Solution: In Section 2.2, we propose an organ-centric pathophysiology-based modeling paradigm. We model the changes in patient state according to the pathophysiology of affected organ systems. Different diseases may impair one or more of the physiological functions performed by different organ systems. Pathophysiology describes the process of such changes in physiological measurements and organ states throughout disease progress. Multiple pathophysiological processes must often be tracked for each disease. Each pathophysiological state encodes the subset of relevant clinical data required for monitoring the patients state/ assessing patient's response to a treatment for that specific state.

Challenge 1.c: The clinical models must be easily understood and validated.

Proposed Solution: Physicians are taught body organ systems and structural relationships within the body. In addition, the pathophysiological description of disease processes is how medical knowledge is commonly known by the physicians [6, 7]. Therefore, the developed models would be at the same abstraction level that medical knowledge and reasoning are known to physicians. This can make clinical validation easier and more effective. To check if our model correctly encodes the medical knowledge, physicians may directly look at the organ models and cross reference with the requirements. In addition, summarizing the patient data in the form of pathophysiological states leads to easier cognitive management in realtime. This can also address the first contributing factor to slips and lapses, that is the lack of integrated and comprehensive patient information.

Challenge 1.d: The modeling paradigm must be applicable to different diseases in acute care. The medical knowledge representation and modeling methodology must be able to model different diseases in acute care. To truly evaluate the applicability of a modeling paradigm, different diseases or acute care scenarios must be studied.

Proposed Solution: We partially address this challenge by applying the proposed modeling to two case-studies. In Chapter 3, we present cardiac arrest case study and in Chapter 4, we discuss modeling ***sepsis***, which is a life-threatening organ dysfunction caused by a dis-regulated host response to infection.

Challenge 1.e: The clinical models must have the correct level of abstraction, concurrency and complexity. Some clinical information, although relevant, may not be needed in the same details as others. On the other hand, the model may need to represent dynamic clinical information and track concurrent state changes. Furthermore, while encoding the relevant information the model must remain manageable.

Proposed Solution: In Section 2.3, we present a set of modeling guidelines to specifically address these challenges. We discuss identifying the concurrent pathophysiological processes to be encoded in the model. We then provide some guidelines on defining the states and transitions for each process as well as the interaction between different processes. We demonstrate how following the guidelines can lead to more modular representation, which allows some of the model components to be studied, analyzed and validated independently of the others.

1.2.2 Additional Design Challenges

Challenge 2: The translation of medical texts into executable models must be clinically correct. Clinical guidance systems can greatly affect patient safety, and therefore, it is critical to validate their correctness. However, translating from the usual text approach for communicating knowledge to an executable structure appropriate for computer processing can be complex. This makes it difficult to ensure the correctness of the encoded medical knowledge in the models.

Proposed Solution: This is addressed by clinical simulation to be validated by physicians. In Section 2.7 we discuss the generic approach we have taken for validating the models. More specific details on validation of the integrated organ-centric pathophysiology-based cardiac arrest models are presented in Section 3.4. We use Yakindu Statechart Tool (SCT) [8], to develop organ and guideline models. This is a tool that provides an integrated modeling environment and allows simulation of the developed model. We have integrated the developed models with an existing guidance system interface, developed by our group. This allows us to simulate different clinical scenarios and validate the dynamic behavior of the developed models.

Challenge 3: The desired behaviors must always hold, not just in the limited clinical simulation. In addition to validating the clinical correctness of the developed models using simulation, it is critical to formally verify the correctness of the models.

Proposed Solution: In Sections 2.5 and 3.2, we address this problem by translating the Yakindu models to UPPAAL [9] for verification. We specify the set of clinical and system properties in linear temporal or computation tree logic. We then use the UPPAAL model checking tool to verify the satisfaction of the specified properties against the models. Currently the translation is done manually and the *correctness of translation* is a challenge to be addressed in the future.

Challenge 4: Physician's deviation from best practice guidelines must be tracked and recorded. At times patient states presented by our system may differ from physician understanding and they may deviate from the best practice guidelines suggested by our system. A source of this inconsistency, should it occur, is that most guidelines are for average population and there may be information used by the physician to make a determination but not available to our system.

Proposed Solution: For each pathophysiological state machine we develop a physician belief state machine, with same static structure. The physician model has the same pathophysiological states; however, the transition between the states are enabled according to physician's response to the patient state suggested by the model. Therefore, the two pathophysiological processes have different active states at run-time, when the physician deviates. We have added a logging interface to the models that allows the deviation instances to be logged. Finally, we propose the divergence-convergence protocol to allow some degree of flexibility, while reducing the unintended deviation from best practice guidelines. When the physician diverges from the system suggestions, the model still keeps track of the patient state according to the best practice guidelines and the corresponding alerts are generated if the patient state deteriorates. However, if the patient state improves the model converges with the physician's decision. We present our physician model in Section 2.4.1.

Challenge 5: The developed models must be integrated into the guidance system and derive its execution.

Proposed Solution: In Section 2.6, we propose a model-driven approach to integrate the developed models with a guidance system, developed by our group. The models interact with the guidance system interface using our handwritten communication code. The developed pathophysiological models then derive the execution of guidance system. This allows us to design alert systems that are integrated with patient's pathophysiological states. This is further discussed in the next design challenge.

Challenge 6: An alarm management approach is required to reduce slips and lapse caused by inadequate alarm configuration and practices. The critically-ill patients' measurements will be out of range for most the time and alarms designed based on only single out-of-range measurements can result in alarm fatigue and increase the possibility of medical errors. An alarm approach that is comprehensive and integrated with patient (pathophysiological) state is critical to address this issue. Additionally, the effectiveness and safety of any proposed alarm management system must be evaluated.

Proposed Solution: In Section 2.4.2, a best practice-defined alert system is proposed, which uses the concept of *actionable alert*. An alert is only generated when a new monitoring and/or treatment action must be performed by medical staff. This often occurs when disease and/or organ state is transitioning towards or has transitioned to a worse stage as defined by the medical best practice guidelines. Furthermore, the alerts contain some contextual information which is the pathophysiological state of the involved organ systems. This is different from normal medical alerts that signal the physiological signals being out of a predefined range; such alerts are not closely coupled with best practice guidelines and may be less useful or even distracting.

To evaluate the effectiveness of our alert system, we have conducted a preliminary clinical validation of cardiac arrest guidance system in a Carle ACLS training class. In Section 3.4, we present the information gathered and discuss some of the lessons learned. There is also a 2-part study protocol with Carle hospital in place. These studies will assess how much improvement our cardiac arrest guidance system can make in terms of adherence to ACLS workflow, compared to traditional approach of memorizing workflows and, using paper and pencil for recording resuscitation events.

The primary contribution and focus of this thesis is addressing the above modeling and design challenges. However, our research also contributes to the following:

Best practice guided runtime device configuration: Medical systems need to inherently be runtime configurable. In acute care environments, the patient condition can change rapidly and as a result of such changes, new devices may be plugged in or an existing device may be removed from clinical configuration. Such dynamic configuration is not seen in other monitoring systems. The safety and efficacy of patient care in a dynamic clinical environment depend on continuously safe use of medical monitoring and therapeutic devices. In Chapter 6, we propose a device configuration management approach that uses the organ-centric pathophysiology-based disease model and patient state representation. We show that a subset of best practice device configuration safety requirements can be described as a device constraint which must be satisfied for specific organ states.

Safety analysis of medical device plug-n-play (MD PnP) systems: MD PnP is a recent NIH (National Institute of Health) sponsored investigation on design of safe integrated medical systems [10]. One goal of this program is adoption of medical device interoperability to significantly mitigate preventable medical errors by developing the technological foundation to design the next generation medical systems. In Chapter 5, we provide guidelines to assist users to safely use an MD PnP system consisting of a commercial platform not designed for safety critical applications. The MD PnP system is said to be safe for an intended medical application, if none of the MD PnP components can become a causal factor of the known safety hazards when the medical devices are functioning according to their specifications. We propose a safety analysis procedure based on extended fault trees to determine whether any of the MD PnP components can become a causal factor of the known safety hazards for the intended medical scenario. We analyze the MD PnP safety for several clinical scenarios.

1.3 Related Work

The *preventable* medical errors are often caused by slips and lapses, which include, but are not limited to delayed diagnosis, delayed or ineffective therapeutic interventions, and unintended deviation from the best practice guidelines. Due to factors such as selective attention[1], time pressure in acute care, and mental overload, medical staff might overlook some of the critical changes in the patient state and this may result in delayed diagnosis and treatment. Updating and displaying the set of monitoring and treatment guidelines as the patient state changes may alleviate this problem.

Many of current medical monitoring devices and systems provide single (physiological) measurement-alerts and thus little contextual information to medical staff. Some research has been done on exploiting the relations between measurements and the development of smart alarms [12], [13]. However, we believe that an integrated computerized system is necessary to help medical staff keep track of critical changes in patient state and perform treatments in a timely manner. In [14], the authors discusses a task-driven guidance systems to reduce preventable medical errors in cardiac arrest resuscitation.

[1]Selective attention is simply the act of focusing on a particular object for a period of time, while simultaneously ignoring irrelevant information that is also occurring [11]. For example, in a cardiac arrest case, physician can be focused on stimulating the heart by administering medications such as epinephrine, while overlooking the fact that the patient is developing renal insufficiency that can be exacerbated if the same heart stimulating drugs are administered with the same dosage.

Treatment validation and workflow adaptation for changing patient state are studied in [15] and [16].

The authors in [17] and [18] provide a comparison study of several different computer-interpretable guideline (CIG) models, including PRODIGY [19], EON [20], GLIF [21], GUIDE , PROforma [22] and Asbru [23]. Each approach focuses on addressing a particular modeling challenge. The comparison was made according to different dimensions that capture the structure of each model. One of the main dimensions of comparison is representation and organization of guidelines. Most of these models encode the guidelines as plans of *decision and actions tasks* and organized into *task networks* and provide nesting mechanisms to simplify top-level plans and support the reuse of of sub-guidelines. Another dimension is the decision model and criteria expression language. Some use switch constructs or mutually exclusive expressions in preconditions. In some models argumentation rules are used to express preferences for/against alternatives candidates of non-deterministic decisions (where more than one alternative is considered appropriate). Decision trees and influence diagrams are also used to represent non-deterministic choices to help identifying the alternative most likely to reach the goal. Medical concepts are often modeled using classification hierarchies, definition of abstract terms and concepts, and relationship between the concepts that convey medical knowledge. Concepts are often defined through mapping to terminology systems.

The comparison dimension, which is of special interest to our work in this thesis is *patient information model*. Often in these works, the main focus is on representation and modeling of guidelines rather than the patient state. PRODIGY, EON, GLIF use a specific modeling component, called Scenario or patient state components to facilitates a patient's automatic entry into an appropriate plan. Other models use expressions of patient states and decision criteria or as preconditions affecting the control flow. A presence criteria is modeled by giving explicit definitions of terms to be checked and determining a method for looking for the terms in the local EMR. Plan selection is based on satisfying a precondition defined in terms of patient state but it can be further improved by matching the effect of plan to target state.

The patient information model in these guidelines systems, focuses on representing the patient data and mapping it to the corresponding items in hospital EMR. This can limit the definition of patients state to the terms used in the local EMR. The representation methodologies often organize relevant information in a single structure. Some, such as PROforma, GUIDE and Asbru, do not put any constraints on the definition of complex objects; others define a specific set of patient data structures and complex objects. PRODIGY and EON use virtual medical record (VMR) abstraction to identify the subset of patient data, from medical records, which is required for decision support applications. These abstractions include diagnosis, test result, medication authorization, procedure, etc. An object-oriented VMR would ease the process of developing guidelines. The criteria and patient states are defined in guidelines by referencing the VMR rather than specific EMR data item.

GLIF structures the patient data according to HL7 reference information model (RIM) [24], [25]. Having a patient information model consistent with HL7-RMI could facilitates mapping guidelines terms to EMRs and interoperability. HL7 Reference Information Model (RIM) is a key component of the HL7 family of standards. It serves as the source from which all other HL7 information models receive its meaning. RIM outlines the logical form of the data and allows establishing the semantic interoperability among different domains in healthcare. It is *"an abstract model which defines the grammar of a language for information in healthcare"*. The "entities" are related to each other through "roles" and their participations in "acts". The model allows the formation of networks of logically or structurally related "acts". Through these networks, composition, reason, order fulfillment can be expressed. The HL7-RIM does not define disease processes, which is among the ultimate subject of health information.

However, mapping guidelines data items to EMRs can still be defined by a system, even if the representation of patient state does not come from the classes and abstracts supported by EMRs, which is mostly based on HL7-RIM. Our organ-centric models may create monitoring and treatment guideline data items, which do not correspond to any specific items in EMRs. However, the subset that has corresponding data items can be mapped. We have integrated the organ-centric pathophysiological models and best practice guidelines models into a guidance system. Our team has started developing an HL7 module for the guidance system that defines the mapping of monitoring and guideline terms codified by our models.

Some of these approaches, specifically the ones using the concept of scenarios, organize the guidelines as a collection if clinical contexts, which are (partial) characterization of patient state. The other approaches support a similar functionality by using expressions that refer to patient states in decision criteria or preconditions to affect the guideline control flow. Some argue the latter approach allows guideline modeling to remain *task-based*, rather than *state-based*. However, clinical scenarios are intuitive and domain expert find them useful. We believe that our pathophysiology-based patient model belongs to the group of state-based approaches. Physicians are taught body organ systems and structural relationships within the body. Moreover, the pathophysiological description of disease processes is how medical knowledge is commonly known by the physicians [6, 7]. Therefore, the developed models would be at the same abstraction level that medical knowledge and reasoning are known to physicians. This can make clinical validation easier and more effective. Physicians may directly look at the organ models and cross reference with the requirements. However, we still can support the use of patient-state based decision criteria and preconditions.

A great body of work in this domain has been focused on rule-based systems and expert systems developed using artificial intelligence. These systems are designed to provide medical diagnosis and treatment suggestions by codifying experts' knowledge, reasoning using the rules and deducing new knowledge [26]. Several inference and decision making methodologies have been developed, such as rule-based reasoning [27], fuzzy logic [28] and probabilistic network [29]. Approaches based on machine learning aim to excerpt common patterns from empirical medical data and make decisions based on the learned behaviors [30]. In [31], a modular representation is proposed, where each guideline is modeled as a Medical Logic Module (MLM) that makes a single decision. This approach focuses on simple independent guidelines. However, most clinical problems are complicated and involve many dynamic decision-making steps; therefore, straightforward attempts to chain together larger sets of rules encounter major difficulties [32]. Furthermore, the medical knowledge are often represented as a set of production rules. Any potential dependency between the rule can not represented and a plan of monitoring or treatment actions unfolding over time can not be represented with individual rules.

Our computational approach codifies patient information into executable organ automata.The main reason behind developing organ-centric disease models is that medical literature and training represent disease as a set of abnormal organ states. There have been similar works on organization of patient information. The authors in [33] propose a framework of open-source services dynamically configurable to transform EHR data into standards-conforming, comparable information suitable for large-scale analyses, inferencing, and integration of disparate health data. In [34] a new application system is proposed to store and categorize patient information according to body system view. However, the focus of these works have been on static information structures and not monitoring rapidly changing patient state. We focus on developing executable disease models that can be directly used to derive the best practice assist system.

In [35], the authors propose a method for medical knowledge representation based on causal associational network. Different observations of the patient is associated with pathophysiological states describing the progress of the disease under study. The relationship ship between different pathophysiological states are represented using the notion of cause and effect. The links between the states are annotated with a likelihood number indicating the strength of causation. It should be noted that the states are not mutually exclusive. The proposed approach, in fact, models the disease as a belief network that is used to analyze the possible causes of an already confirmed diagnosis. At the start, the set of confirmed pathophysiological states representing the patient current disease are identified. Then they determine the most likely starting states, from which most likely pathways are generated that traverse through the greatest number of confirmed states. The generated pathways represent the most likely explanations of the disease processes. The physician then uses these pathways to classify the disease based on its mechanism and the severity and the degree of progression. A treatment plan can be associated with each diagnosis class. In our proposed approach, we dynamically track the state of different pathophysiological processes. Furthermore, we take a more modular approach and model the parallel pathophysiological processes using concurrent state machines. Our executable models are then used to monitor the patient and concurrent changes occurring as a result of the disease. Maintaining a large belief network can become challenging. Furthermore, developing the model for a more complex disease involving a larger number of pathophysiological states can be non-trivial. Even making modification to the same disease model may require propagating the changes throughout the whole model. The modularity becomes critical when the model needs to be modified, validated or formally verified. Moreover, in our approach some of the pathophysiological processes can be re-used in different disease.

Another body of work has been focused on formally defining medical processes. The documents describing different medical protocols used to administer healthcare, may be incomplete and include terms are either poorly defined or used inconsistently throughout the document. Moreover, important details, specifically the ones related to handling exceptional cases, might be missing. The use of such defective processes for providing health care services can contribute to medical errors that ultimately affect patient safety. The authors in [36, 37] focus on tackling this problem by proposing an approach to formally define medical processes. The authors use Little-JIL process definition language to specify different recommended sequence of actions to perform a specific clinical task. The precisely defined model of medical processes, can then be used for rigorous analysis.

Little-JIL originally developed to define processes that coordinates the activities of multiple agents [38]. The main component of the language is the coordination specification, which has a graphical representation but also its syntax and semantic are precisely defined. The central abstraction in the language is the *step* and facilities for specifying iteration and concurrency is included in the language. We believe the authors use a task-based modeling approach, similar to the guidelines model discussed above. The analysis of formally-defined human-intensive medical processes is further discussed in [39, 40]. The paper describes using finite state verification to find defects in the process definition or the medical processes itself. This is achieved by determining if the specified properties are always satisfied on all possible execution paths through the process definition; but it is assumed that all agents involved in the process perform their tasks without error. The authors have built tools to automatically translate Little-JIL process definition to the input representation of verification tools such as SPIN [41]. Each property is represented as a finite-state automaton that describes the acceptable or unacceptable sequence of selected events in the process being analyzed. Human-intensive medical processes, such as blood transfusion and chemotherapy, are used to demonstrate the value of the proposed approach. Several errors, including possible deadlock, was discovered in these processes.

In [42] the use of other static analysis technologies, such as Failure Modes & Effects Analysis (FMEA) [43], and Fault Tree Analysis (FTA) [44, 45] for medical process improvement is discussed. The authors apply FMEA to formally defined processes to evaluate the impact of individual step failures on the whole process. This technique can demonstrate the propagation of a single failure and the ultimate effect which often indicate a safety hazard directly and explicitly violating patient safety. FTA analysis can also be used to study all the failure modes leading to a specific safety hazard. In addition to static process analysis, the authors also discuss the simulation technique that uses the dynamic process execution state, specifications of the behaviors and capabilities of agents, and a resource model .The technique is used to study a different range of problems, such as effect of resource (i.e. the doctors, nurses, and beds) allocation on the length of patient stay.

The formal process models proposed in [39] is also used in [46] for online deviation detection in medical processes. The main goal in this work is to reduce medical errors by encouraging conformance with best practice guidelines and recommended processes. The authors adopt the definition of medical errors from the IOM report [2] and focus on ”error of planning”, which is the use of wrong plan to achieve a goal (vs. error of execution, which is the failure of a planned task to be performed as intended). Examples of planning errors are omitting a required task or performing a task that should not have been done. In the proposed approach, process execution is tracked and the sequence of performed steps is continually compared with all the valid and recommended sequence of steps to detect deviations. After the detection phase trace selection is used to hypothesize the locations of potential errors. A set of chosen traces are likely candidates the process performer had intended to carry out and are more similar to the deviant. The difference between each selected trace and the deviant then can be used to pinpoint the location of the potential error in the sequence of performed steps in the corresponding medical process. It should be noted that delayed deviant detection might occur. A preliminary analysis is presented but, as discussed by the authors, this issue needs to be investigated for more complex and realistic medical processes. The scalability and applicability of this approach need to be also studied for more realistic scenarios then the one described and used in this paper. The steps need to be logged manually by the human operators until the technology for automatic recognition and recording of different steps of a medical process as they are accomplished. Similarly, in our proposed physician model we also assume the physician response to the system-suggested guidelines to be manually entered into the system. However, the (monitoring and treatment) steps in our model are more coarse grained compared to the steps described in these human-intensive processes model. This reduces the number of interactions required between the system and medical staff.

In [47], the authors present a conceptual architecture and the necessary system components for aiding the effective execution of human-intensive medical processes. [48] further describes how the formally-defined and previously analyzed processes can be used to create context-based checklists to improve the outcomes in human-intensive healthcare systems. In addition to a process model, the proposed *Smart Checklist Generator* also uses a the events generated by a process execution monitor. The checklist is continuously updated as the health care personnel go through the different steps of the process. Some examples of events of interest are the completion of a subtask, occurrence of an exception during the process execution or receipt of a patient data sent by medical devices or entered manually. A visualization is provided for different aspects of the current execution of the process. This includes the tasks that have been successfully performed, the current tasks being executed and the possible future steps. Each of these items in the smart checklist can be annotated with necessary and important information, such as needed artifacts, the performers responsible for finishing the task and possible exception that can occur when performing the task. We also believe the use of context information to dynamically adapt the checklist can improve the efficiency these checks. Moreover, it can alleviate the problems caused by long, complex and generic checklists.

The focus of this body of work is on human-intensive clinical protocols used for providing different healthcare services. The proposed medical process modeling approach is mostly used for offline static analysis of medical processes. Only preliminary work on the realtime use of formally defined medical processes to derive the execution of guidance systems has been proposed. Real-time constraints are a critical aspect of any medical system. However, Little-JIL provides little and primitive support for timing constructs. This makes the specification and analysis of both hard and soft real-time constraints challenging. Furthermore, maintaining and updating process models for more complex clinical protocols can be challenging and a more modular representation may be beneficial. Finally, medical knowledge representation regarding patient state is not discussed in these works. However, we can use and benefit from the proposed approach if we decide to further focus on the coordination aspect of the human-intensive processes used to deliver treatments from best practice guidelines. This includes management of interaction of health care providers with each other and with the patient.

The clinical guidance systems must integrate clinical data form different sources, mainly medical devices. Therefore, interoperability of medical devices is critical for designing the system [49]. The recent investigation sponsored by National Institute of Health (NIH) on safe medical system reconfiguration and interoperability, Medical Device Plug-and-Play (MD PnP) [10], focuses on developing the technological foundation to design the next generation medical systems. The Integrated Clinical Environment (ICE) standard is one such development, which is described in ASTM 2761-09(2013) [50]. This standard describes the logical elements of an MD PnP system but does not specify implementation details. The MD PnP system supports a range of safe medical system reconfigurations. The medical devices are plugged in as needed, information from different sources are integrated, and medical actions are coordinated to provide an integrated clinical environment. To facilitate the development of medical device interoperability, the MD PnP program has developed an open source implementation of the Integrated Clinical Environment and made it freely available on SourceForge [51]. The platform consists of software device adapters for medical devices (including anesthesia machines, ventilators, and patient monitors), Object Management Group (OMG) Data Distribution Service for Real-Time systems (DDS) standard middle-ware, and demonstration applications.

The necessity of proper infrastructures and the principles of medical platforms have also been discussed in [52, 53]. Advanced monitoring systems, such as the proposed assist system in this thesis, can benefit from the technological foundations such as MD PnP framework and built upon it. There are several works, which have also studied medical device interoperability to improve patient safety. In [54], the authors discuss a low complexity coordination architecture for networked supervisory medical systems. In [55, 56], a coordination framework is introduced to facilitates integration of different medical devices, and provide a model-based app programming environment for easy assembling of a new medical application . In [57] the authors report their experience with a prototype plug-and-play synchronized medical system including an x-ray and a ventilator. An interoperable medical system including a patient controlled analgesia (PCA) pump is the focus in [58, 59]. While these works focused on a static configuration, our focus has been on a more generic scheme to ensure patient safety in the more dynamic configuration of acute care.

Chapter 2

Organ-Centric Pathophysiology-Based Modeling Paradigm

A main prerequisite for developing a guideline system is creating computer interpretable representations of he clinical knowledge contained in best practice guidelines. In other words, creation of such systems requires considerable amount of **modeling** activity such as deciding what patient data is relevant and identifying the concept and relationships among concepts that relate to guideline generations [5]. The main focus of this thesis is on **developing executable clinical models for acute care**.

We propose an organ-centric pathophysiology-based modeling paradigm, where we translate the medical text into executable interactive disease and organ models. We model the patient state and different disease stages according to pathophysiology of organ systems, which describes how each of the organ functions are affected by the disease. As mentioned before, such clinical models are studied extensively in medical community. However, the common format used to represent and communicate clinical knowledge, including such models, is English (*text and graphics*). Therefore, utilizing these models when providing clinical care can be as challenging as using a map while driving. Turning maps to GPS based navigation tools has revolutionized driving. Translating pathophysiological models, based in medical best practice text, into executable models can have a similar effect on the use of computer guidance systems in medicine.

However, there are several challenges for developing executable clinical models that must be addressed. In the rest of this chapter, we first discuss these modeling challenges in Section 2.1. Next, we describe how we address each challenge by proposing an organ-centric pathophysiology-based modeling paradigm and introducing a set of modeling guidelines in Sections 2.2 and 2.3, respectively. The next step is applying the guidelines to construct he organ-centric pathophysiological models. However, the description of this step is postponed to Chapters 3 and 4, when we discuss our two cases studies in more detail. We believe the model construction procedure can be described by case studies more clearly and effectively. We discuss the general approach we have taken to formally verify the model's safety properties in Section 2.5. We present a model driven approach for integrating the models into a guidance system in Section 2.6. Finally, the clinical validation of the models are discussed in Section 2.7.

2.1 Modeling Challenges

In Chapter 1, we presented a list of the modeling challenges we have identified in this thesis. In this section, we present these challenges in greater details. We refer to $\mathbf{ith}$ modeling challenge as $\mathbf{MCH_i}$:

$\mathbf{MCH_1}$: **Medical knowledge representation must be in a *machine-processable form*.** The medical knowledge must be translated from text format to a structure appropriate for the logical application of that knowledge by a computer guidance system. The developed clinical model(s) should be executable or must provide a means for automatic execution to facilitate the design of best practice guidance systems.

$\mathbf{MCH_2}$: **The developed models must encode the relevant clinical information.** The model needs to encode the medical information or knowledge relevant to the disease under study. For example, a physician often is interested in not only diagnosis itself but also the process of diagnosis and the clinical data required to make a diagnosis. However, the encoded information must be adequate but not excessive such that it create a mental overload.

$\mathbf{MCH_3}$: **The medical knowledge representation and other encoded clinical knowledge must be easily understood and validated.** Physicians can play a critical role in development and validation of executable clinical models. Therefore, models must encode the relevant medical knowledge *at the same abstraction level that medical knowledge and reasoning are known to medical domain experts.*

$\mathrm{MCH_4}$: **The modeling paradigm must be applicable to different diseases in acute care.** The medical knowledge representation and modeling methodology must be able to model different diseases in acute care. Initially, we have used the cardiac arrest case study when developing our model. However, to evaluate the applicability of our proposed modeling paradigm, different diseases or acute care scenarios must be considered.

$\mathrm{MCH_5}$: **The correct level of abstraction must be chosen for each model.** Some clinical information, although relevant, may not be needed in the same details as others. Therefore, to develop efficient and effective models, we must identify the correct level of abstraction for different modeling components.

$\mathrm{MCH_6}$: **The model must have the correct level of concurrency.** The model may need to represent dynamic clinical information and may be used to track different aspects of patient state that may change simultaneously. Therefore, the model must be designed in a such way that it would be able to concurrently track all the critical changes.

$\mathrm{MCH_7}$: **The model complexity must be managed.** While encoding all the relevant information, the model must remain manageable. This is critical since development and modification of highly complex models can be error-prone. Furthermore, the formal verification task may grow more challenging as the model complexity increases. This becomes specifically useful when modeling complex diseases involving high number of pathophysiological processes.

In the next section, we present our organ-centric pathophysiology-based paradigm and specify the subset of modeling challenges addressed by the proposed paradigm. We address the remaining ones by introducing a set of modeling guidelines, later in the chapter.

2.2 Modeling Paradigm

We propose an organ-centric pathophysiology-based modeling paradigm, where patient state is modeled according to pathophysiology of affected organ systems. This is possible because disease and organ states are already discretized by the medical community; for example, different stages of cancer, diabetes, heart diseases, etc. Furthermore, the diagnosis and treatment are organized for each stage.

Different diseases may impair one or more of the physiological functions performed by different organ systems. Pathophysiology describes the process of such changes in physiological measurements and organ states throughout the disease progress. Multiple pathophysiological processes must often be tracked for each disease. Therefore, an executable organ-centric pathophysiology-based model would encode the key clinical information relevant to the disease under study. This partially addresses the modeling challenge $\mathbf{MCH_2}$. However, additional considerations are required when designing the specific modeling components to address this challenge more efficiently. We further discuss this when presenting our modeling guidelines in Section 2.3. The following summarizes the main reasons behind developing our models according to this paradigm:

- Physicians are taught body organ systems and structural relationships within the body. Moreover, the pathophysiological description of disease processes is how medical knowledge is commonly known by the physicians [6, 7]. *The developed models would be at the same abstraction level that medical knowledge and reasoning are known to physicians*. They may directly look at the organ models and cross reference with the requirements. Consequently, choosing an organ-centric pathophysiology-based paradigm to represent patient and disease state can make clinical validation easier and more effective. In other words, the proposed modeling paradigm can address the challenge $\mathbf{MCH_3}$.

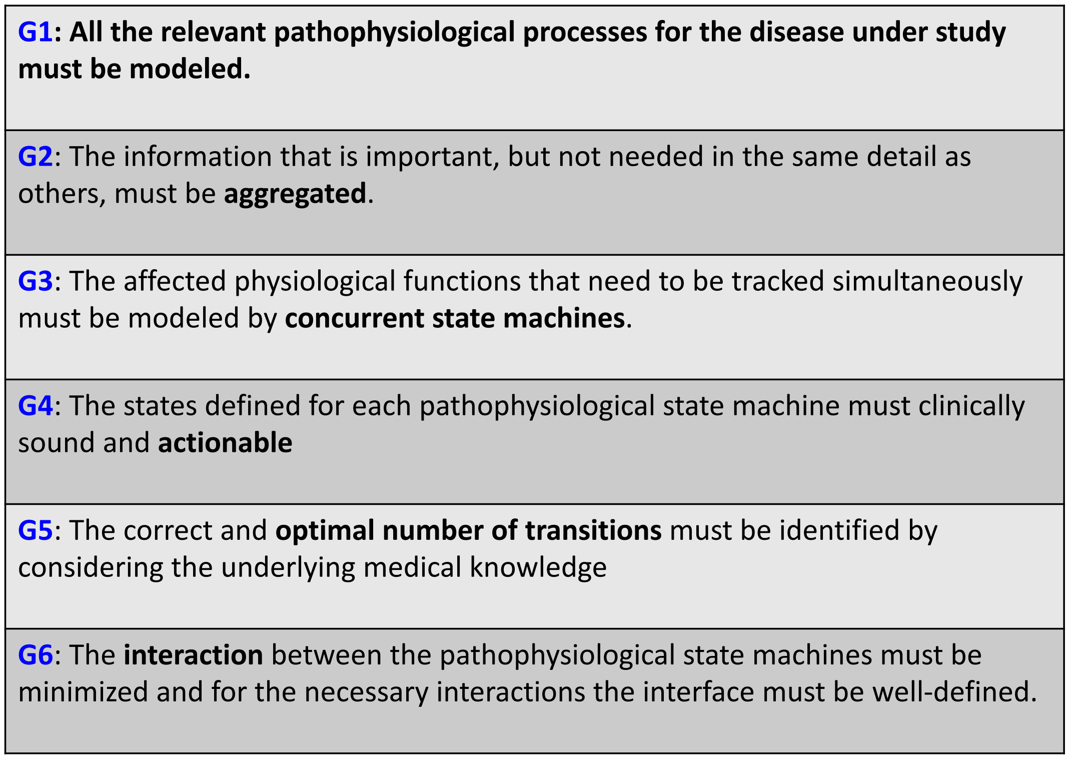
- Multiple organ systems may be affected by a disease and the interaction among different organ systems, and monitoring and therapeutic actions make the dynamics very difficult for physicians to follow. Therefore, computational methods that collect, organize, and process patient information according to organ-based pathophysiological models and presents them in a timely manner can serve physicians in the best way possible.

- Physicians often describe *monitoring and therapeutic requirements for different organ states*. For example, ACLS algorithm indicates that *shock* must be considered when shockable rhythm is confirmed. This is a treatment guideline for arrhythmia, which is an insufficiency of cardiovascular organ system. Therefore, the proposed modeling paradigm could also facilitate the design of any clinical guidance system that encodes the best practice monitoring and treatment guidelines.

To develop our models, we use finite state machine formalism. We model a disease according to the pathophysiology of involved organ systems, which describes how each of the organ functions are affected by the disease. Each state machine, which is a labeled transition system, models one such process. The nodes represent different pathophysiological states. The state variables may include physiological measurements and lab values. The definition of each state and the condition enabling transitions between the states are extracted from medical literature and clinical best practice guidelines. These models have formal semantics and are executable. Therefore, they can be used to track disease progress, organ state deteriorations, and organ state improvements, etc. This addresses the challenge $\mathbf{MCH_1}$. We use an open-source tool, called *Yakindu*, for developing and simulating he state machines and integrating them into a guidance system. The details on model integration are discussed in Section 2.6.

We presented a brief description of cardiac arrest in Section 1.1. This disease mainly affects the cardiovascular organ system. The heart rhythm may change and include abnormal or irregular heart rhythms or waveforms (arrhythmias). Furthermore, The insufficient blood circulation in the lungs impedes the CO_2 and O_2 exchange resulting in CO_2 accumulation in the blood, which may be related to a rise in blood acidity level indicated by decrease in blood pH level. The abnormal pH is revealed in arterial blood gas (ABG) test. These are two examples of pathophysiological changes in cardiovascular organ system caused by the disease.

A simplified cardiac arrest model was presented in Figure 1.1. We have developed the *arrhythmia* and *blood gas imbalance* automata among others. In each of these automata we codify different stages/-types of heart rhythm irregularity or imbalance in arterial blood gas levels according to existing medical knowledge. Fo example, *pulseless electrical activity*, a heart rhythm observed on the electrocardiogram that should be producing a pulse, but is not, is a state in the *arrhythmia* automata. Similarly, *severe metabolic acidosis*, a very low blood pH, is a state in *blood gas imbalance* automata. When new blood pressure, heart rate, and EKG readings are received by our best practice system and blood pH value from a recent ABG test is entered by the nurse, the *arrhythmia* and *blood gas imbalance* automata will transition to the appropriate states. Let us assume the current readings indicate a very low blood pressure and blood pH, no meaningful heart rate reading and an EKG rhythm that is not a flat line but does not produce a pulse. Therefore, the *arrhythmia* machine transitions to *pulseless electrical activity* state



G1: All the relevant pathophysiological processes for the disease under study must be modeled.

G2: The information that is important, but not needed in the same detail as others, must be **aggregated**.

G3: The affected physiological functions that need to be tracked simultaneously must be modeled by **concurrent state machines**.

G4: The states defined for each pathophysiological state machine must clinically sound and **actionable**

G5: The correct and **optimal number of transitions** must be identified by considering the underlying medical knowledge

G6: The **interaction** between the pathophysiological state machines must be minimized and for the necessary interactions the interface must be well-defined.

Figure 2.1: The List of Modeling Guidelines

and *blood gas imbalance* transitions to *severe metabolic acidosis*. We present a more detailed cardiac arrest model after presenting our modeling guidelines.

The use of organ-centric pathophysiology-based paradigm and finite state machine formalism to develop our models, only addresses $\mathbf{MCH_1}$and *partially* addresses $\mathbf{MCH_2}$ and $\mathbf{MCH_3}$. In the next section we introduce a set of modeling guidelines to further address $\mathbf{MCH_2}$ and $\mathbf{MCH_3}$ and challenges $\mathbf{MCH_5}$ - $\mathbf{MCH_7}$. We partially address the final remaining challenge $\mathbf{MCH_4}$, that is applicability of the proposed modeling paradigm to different diseases in acute care, by applying the paradigm and guidelines to construct the ***sepsis*** model. In comparison to cardiac arrest, this is a more complex case study an involves more pathophysiological processes.

2.3 Modeling Guidelines

In this section we present the modeling guidelines we have developed to specifically address the challenges $\mathbf{MCH_5}$ - $\mathbf{MCH_7}$. We describe each guideline and explain the challenges it addresses.

Figure 2.1 shows the list of these guideline. Note that there is a *many-to-many* relationship between the identified modeling challenges and proposed guidelines. An illustrative summary of this relationship is provided at the end of this section. The modeling guidelines are as follows; we refer to $\mathtt{ith}$ guideline as $\mathbf{G_i}$:

$\mathbf{G_1}$: **All the relevant pathophysiological processes for the disease under study must be modeled.** As mentioned in the previous section, each disease can cause several abnormalities and insufficiencies in different organ systems. For example, in cardiac arrest, the main affected organ is the cardiovascular system. The effect mainly includes abnormal heart rhythm and blood acidity and gas levels. However, other organs might also be affected by cardiac arrest. For example, while medical staff focus on treating arrhythmia and injecting various drugs, such as epinephrine, patient may develop renal insufficiency. If the medical staff do not notice the new condition and continue giving the same drugs with the same dosage, patient's kidney may become severely insufficient or even fail. Therefore, we must include all the organ and insufficiency category models relevant to the disease under study. In other words, *a pathophysiological process becomes relevant if the corresponding physiological function* ***is affected*** *by the disease and/or* ***affects*** *the treatment plan.*

The relevant pathophysiological processes would be used to track the disease progress, patient's response to treatments or adapt the treatment plans. For cardiac arrest, in addition to arrhythmia and blood gas imbalance state machines, our model also includes a renal insufficiency state machine to demonstrate the multi-organ interactions. Note that identifying these pathophysiological processes is guided by existing medical knowledge and best practice text. *This guideline further addresses* $\mathbf{MCH_2}$*, that is identifying the relevant clinical information to be encoded in the model.* The next few guidelines help us determine what information to encode in each pathophysiological process and how to model them using finite state formalism.

$\mathbf{G_2}$: **The information that is important, but not needed in the same detail as others, must be aggregated.** This is also determined by pathophysiology of the disease under study. Different pathophysiological functions of the main affected organ system(s) are often monitored individually and in great detail. In addition, more measurements are used to assess each of these pathophysiological functions, compared to the other organs. The choice of clinical measurements to encode in each pathophysiological state machine is derived from the best practice guidelines for the disease in cooperation with the physicians.

For example, in the case of cardiac arrest, most guidelines focus on cardiovascular system as the main affected organ. Therefore, more detailed information regarding cardiovascular abnormalities such as arrhythmia, blood gas imbalance and etc. may be codified in the model. On the other hand, the medical information regarding the pathophysiology of kidney organ system are aggregated as degree of renal insufficiency. *This guideline aims at addressing* $\mathbf{MCH_5}$*, that is choosing the correct level of abstractions for each model component.*

$\mathbf{G_3}$: **The affected physiological functions that need to be tracked simultaneously must be modeled by separate finite state machines.** If the modeling guideline $\mathbf{G_2}$, has been already applied, following this one becomes straightforward. Previously we have specified different (aggregated) pathophysiological processes that need to be tracked. These processes are often occur simultaneously in patient's body. To keep track of all these states and prevent any critical changes from being overlooked, we model each process using a separate state machine.

Figure 2.2 shows the concurrent states machines developed for each of our case studies. Each machine encodes and tracks one of the simultaneous insufficiencies caused by the corresponding disease. Th details on sepsis model will be provided in Chapter 4. *This guideline addresses* $\mathbf{MCH_6}$*, that is determining the correct level of concurrency for the model.*

The next three guidelines are introduced to address the modeling challenge $\mathbf{MCH_7}$, that is managing the model complexity.

$\mathbf{G_4}$: **The states defined for each pathophysiological state machine must clinically sound and actionable.** The previous guideline has helped us identify the concurrent state machines that needs to be developed. The next step is to define the states for each pathophysiological state machine. Each state is defined by correlating relevant physiological measurements and clinical data. The choice of state variables are guided by medical literature and/or given by our physician collaborators. [1]

However, defining a large number of states for each machine can make the model very complex and unmanageable. Moreover, not all the changes in physiological measurements are relevant or useful for a given disease. Providing too much information and too many alerts to medical staff may create

[1]There are measurements, which physicians often look at as a group and interpret together to have a better assessment of patient state.

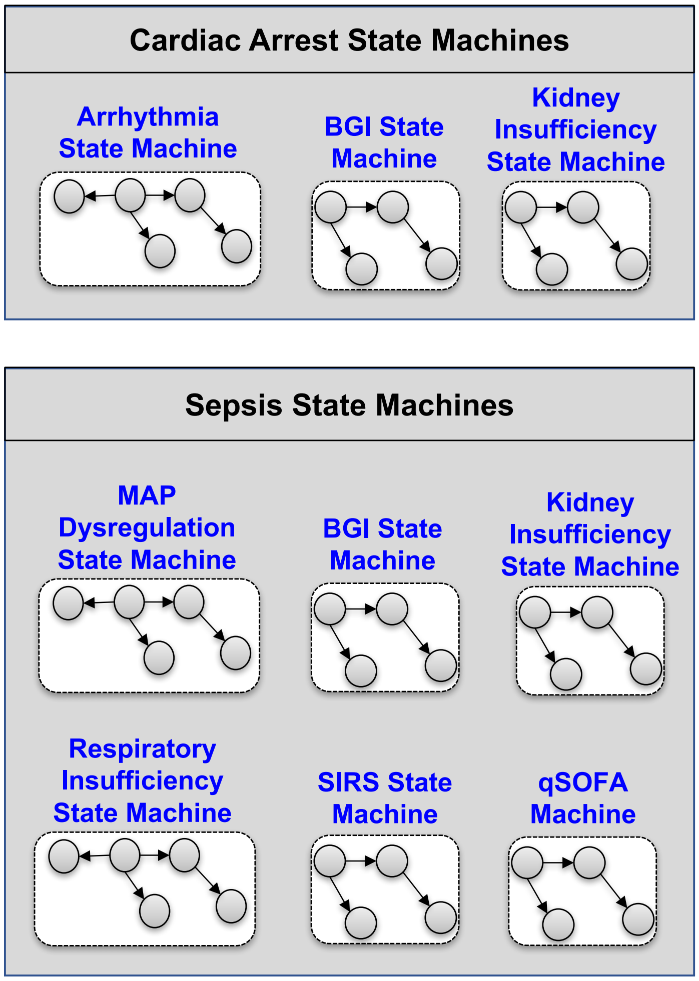

Cardiac Arrest State Machines
Arrhythmia State Machine
BGI State Machine
Kidney Insufficiency State Machine
Sepsis State Machines
MAP Dysregulation State Machine
BGI State Machine
Kidney Insufficiency State Machine
Respiratory Insufficiency State Machine
SIRS State Machine
qSOFA Machine

Figure 2.2: Concurrent Pathophysiological State Machines for Cardiac Arrest and Sepsis Models

confusion and increase their mental overload, which is what our system aims to prevent in the first place. Therefore, we have used the notion of *actionable states*. We define a new state when there is a need for a new monitoring or therapeutic plan. The actionable states can be extracted from best practice or the clinical requirements specified by physician.

Figure 2.3 shows a simplified respiratory insufficiency state machine demonstrating how this guideline can be applied. The states in this machine represent different degrees of respiratory insufficiency and critical change in patient's respiratory state that require an action from medical staff. For example, the setting for ventilation therapy must be adapted when patient transitions to a new state. The transitions indicating organ state improvement are not displayed for better readability.

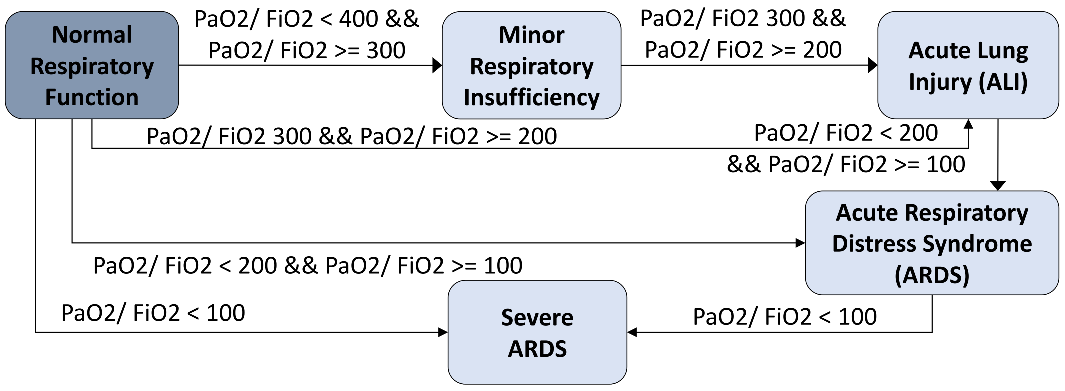

Normal Respiratory Function
PaO2/ FiO2 < 400 && PaO2/ FiO2 >= 300
Minor Respiratory Insufficiency
PaO2/ FiO2 300 && PaO2/ FiO2 >= 200
Acute Lung Injury (ALI)
PaO2/ FiO2 300 && PaO2/ FiO2 >= 200
PaO2/ FiO2 < 200 && PaO2/ FiO2 >= 100
Acute Respiratory Distress Syndrome (ARDS)
PaO2/ FiO2 < 200 && PaO2/ FiO2 >= 100
PaO2/ FiO2 < 100
Severe ARDS
PaO2/ FiO2 < 100

Figure 2.3: ***Actionable States*** **in Respiratory Insufficiency State Machine**

In addition to $\mathbf{MCH_7}$, this guideline also helps further address the modeling challenges $\mathbf{MCH_3}$ and $\mathbf{MCH_4}$. The pathophysiological models with smaller number of states can be more easily understood and validated. Furthermore, the notion of actionable state helps with identifying the relevant changes in patient state that needs to be encoded in the model.

$\mathbf{G_5}$: **The correct and optimal number of transitions must be identified by considering the underlying medical knowledge.** Similar to the definition of states in each machine, defining a large number of transitions can make the model complex and less understandable. Moreover, the process of model development becomes more error prone. The key in identifying the correct and optimal number of transitions is considering the medical knowledge underlying our models.

The required transitions can be extracted from the pathophysiological process being modeled by the state machine. Development and resolution of organ failure may take some time. Therefore, when continuous physiological variables are used to define different insufficiency states, we may not need to create an edge between every insufficiency states. The transitions are only defined to represent the progress of insufficiency. The exception may be the initial state; since we do not know the patient state when we start, we can define a transition between the initial state and every other insufficiency states. This can remove any potential latency in patient state tracking associated with the model stabilization. If we do not have the transition edges from the initial state, when the system starts the models might need a few cycles to stabilize and reflect the current patient state. However, when discrete measurement, such as EKG rhythm, are used to define different states we may require to define transition between all insufficiency states. Figure 2.4 shows renal insufficiency and arrhythmia state machine. The transitions

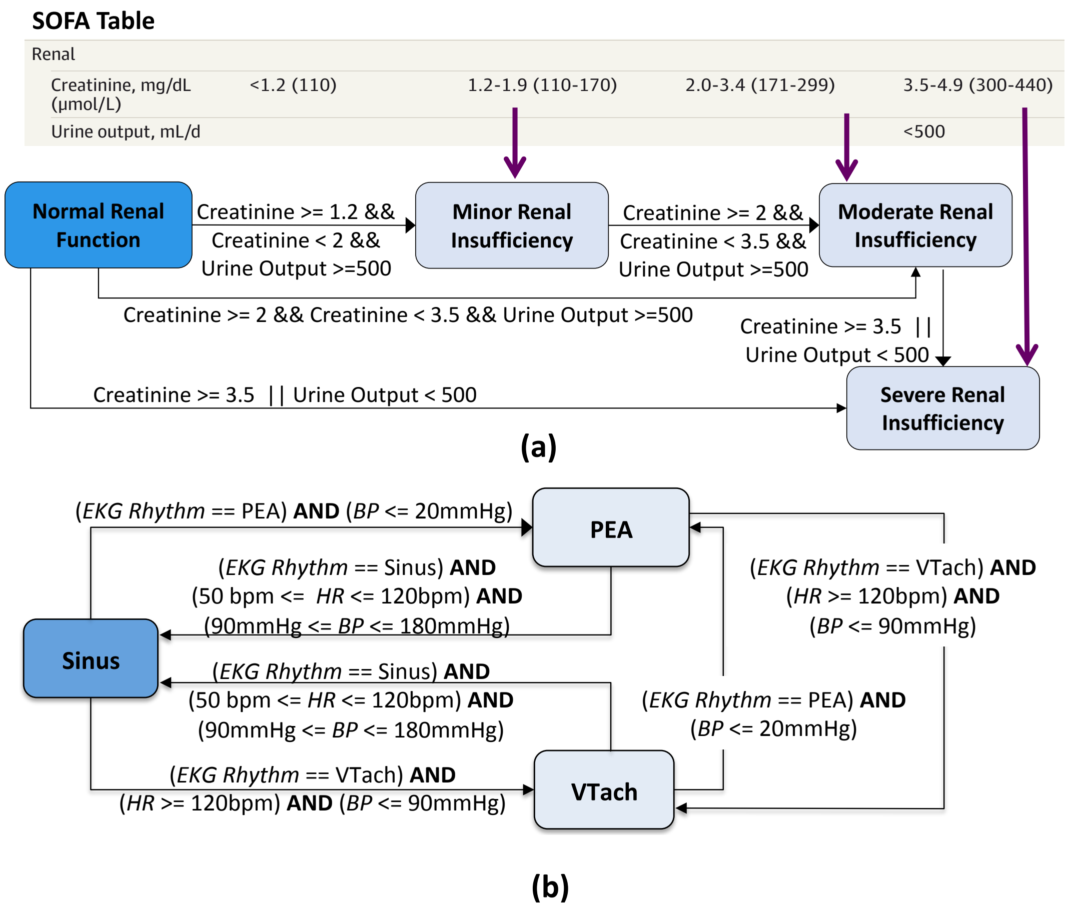

SOFA Table
Renal
Creatinine, mg/dL (μmol/L)
<1.2 (110)
1.2-1.9 (110-170)
2.0-3.4 (171-299)
3.5-4.9 (300-440)
Urine output, mL/d
<500
Normal Renal Function
Minor Renal Insufficiency
Moderate Renal Insufficiency
Severe Renal Insufficiency
Creatinine >= 1.2 && Creatinine < 2 && Urine Output >=500
Creatinine >= 2 && Creatinine < 3.5 && Urine Output >=500
Creatinine >= 2 && Creatinine < 3.5 && Urine Output >=500
Creatinine >= 3.5 || Urine Output < 500
Creatinine >= 3.5 || Urine Output < 500
(a)
PEA
Sinus
VTach
(EKG Rhythm == PEA) AND (BP <= 20mmHg)
(EKG Rhythm == Sinus) AND (50 bpm <= HR <= 120bpm) AND (90mmHg <= BP <= 180mmHg)
(EKG Rhythm == Sinus) AND (50 bpm <= HR <= 120bpm) AND (90mmHg <= BP <= 180mmHg)
(EKG Rhythm == VTach) AND (HR >= 120bpm) AND (BP <= 90mmHg)
(EKG Rhythm == VTach) AND (HR >= 120bpm) AND (BP <= 90mmHg)
(EKG Rhythm == PEA) AND (BP <= 20mmHg)
(b)

Figure 2.4: Pathophysiological State Machines with (a) Continuous State Variables and (b) Discrete and Continuous State Variables

are defined following this guideline. In Renal machine the transitions are only defined to represent the progress of insufficiency, with the exception of initial state.

Transitions are annotated with guards expressing the physiological condition that indicates the destination insufficiency state and only that state. The definition of states in each pathophysiological processes may not be mutually exclusive, according to guidelines they are extracted from. This occurs when a different set of physiological measurements are used to define these states. However, to make the model behavior deterministic we ensure that the guards on all outgoing transitions from the corresponding state are mutually exclusive. 2.4

This improve model understandability and makes clinical validation more efficient and therefore helps address the modeling challenge $\mathbf{MCH_3}$. Figure 2.4 shows the mutually exclusive outgoing

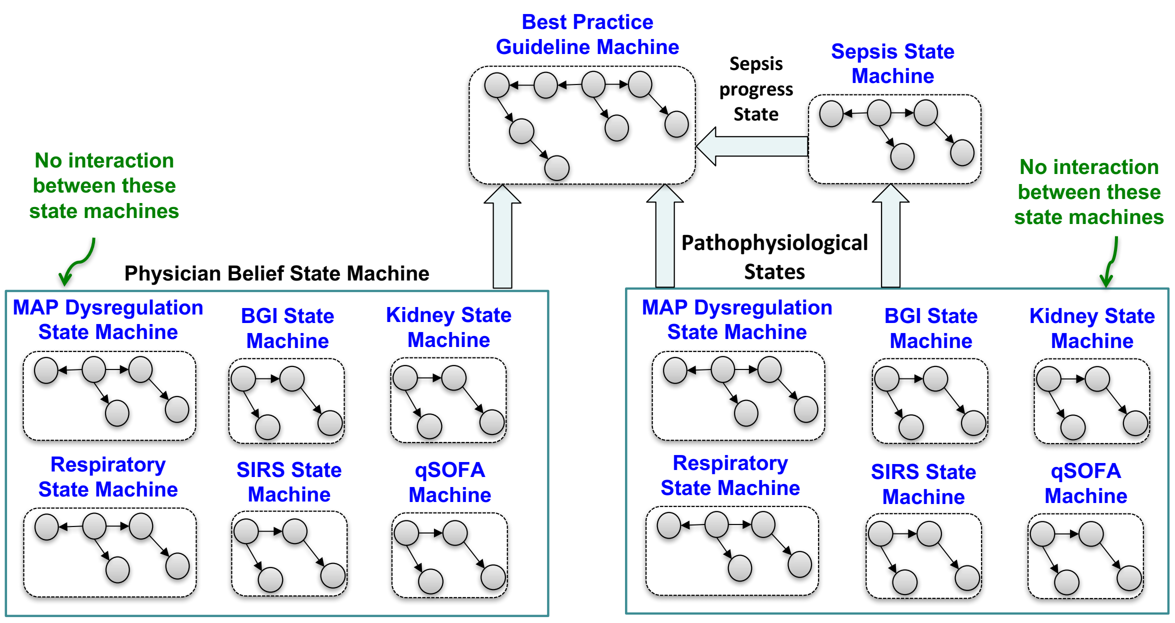

Best Practice Guideline Machine
Sepsis progress State
Sepsis State Machine
No interaction between these state machines
Physician Belief State Machine
Pathophysiological States
No interaction between these state machines
MAP Dysregulation State Machine
BGI State Machine
Kidney State Machine
Respiratory State Machine
SIRS State Machine
qSOFA Machine
MAP Dysregulation State Machine
BGI State Machine
Kidney State Machine
Respiratory State Machine
SIRS State Machine
qSOFA Machine

Figure 2.5: Interaction Diagram for Sepsis Model

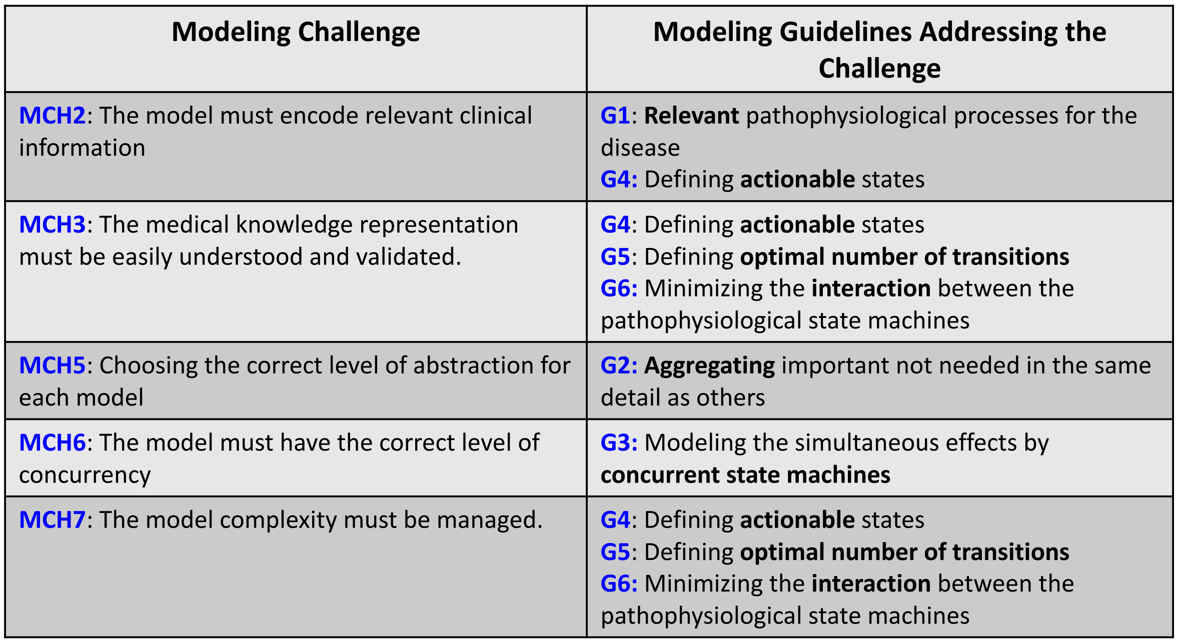
transitions we have defines for each renal insufficiency state. Note that the definition of states, which are extracted from SOFA [2] table, are not explicitly mutual exclusive [60]. The transitions indicating renal organ state improvement are not displayed for better readability.

$\mathbf{G_5}$: **The interaction between pathophysiological state machines must be minimized and for necessary interactions the interface must be well-defined.** Any potential interaction between different state machines can be encoded in a central disease state machine. Figure 2.5 shows the interaction diagram for the sepsis model. The interaction with best practice guideline and physician models will be discussed later.

We try to minimize any communication or dependency between the pathophysiological machines. This improves the *model modularity*. The complexity of validation and verification may be reduced since most of pathophysiological state machines can be validated and verified independently of the others. Lower interaction complexity also results in models that can be more easily understood. Therefore, in addition to $\mathbf{MCH_7}$, this guideline partially addresses the modeling challenge $\mathbf{MCH_3}$ as well.

In this section, we discussed a set of modeling guidelines and the challenges addressed by each one. As mentioned above, some of the challenges may be addressed by multiple guidelines. In Figure

[2]Sequential Organ Failure Assessment



Modeling Challenge	Modeling Guidelines Addressing the Challenge
MCH2: The model must encode relevant clinical information	G1: **Relevant** pathophysiological processes for the disease G4: Defining **actionable** states
MCH3: The medical knowledge representation must be easily understood and validated.	G4: Defining **actionable** states G5: Defining **optimal number of transitions** G6: Minimizing the **interaction** between the pathophysiological state machines
MCH5: Choosing the correct level of abstraction for each model	G2: **Aggregating** important not needed in the same detail as others
MCH6: The model must have the correct level of concurrency	G3: Modeling the simultaneous effects by **concurrent state machines**
MCH7: The model complexity must be managed.	G4: Defining **actionable** states G5: Defining **optimal number of transitions** G6: Minimizing the **interaction** between the pathophysiological state machines

Figure 2.6: Summary of Modeling Challenges and Guidelines

2.6, we show this many-to-many relationship between the identified modeling challenges and proposed modeling guidelines.

2.4 Additional Model Components

In addition to the modeling challenges, we presented additional design challenges in Chapter 1. In this section we specifically focus on the following:

- **Challenge 4: Physicians deviation from best practice guidelines must be tracked and recorded**
- **Challenge 6: An alarm management approach is required to reduce slips and lapse caused by inadequate alarm configuration and practices**

The role of human in the medical system can be more critical compared to most of other *cyber-physical human systems (CPHS)*. Moreover, the complexity and uncertainty of medicine sometimes makes definitive decisions difficult. Therefore, it is critical to account for the human (specifically, physician) as an important aspect of the system. In Section 2.4.1, we propose a compositional model and a protocol to record and track physician response to the information provided by our model (system). In Section 2.4.2, as part of our guideline model, we present an alarm approach, which is integrated with pathophysiological states.

Pseudocode 1 Convergence-Divergence Protocol

```
input 1: $OrganAutomata$ $\textbf{\textit{OA}} = (OA_1, OA_2, \ldots, OA_n)$
input 2: $PhysicianAutomata$ $\textbf{\textit{PA}} = (PA_1, PA_2, \ldots, PA_n)$
input 3: $PatientState$ $\textbf{\textit{S}} = (S_1, S_2, \ldots, S_n)$, input 4: $PhysicianBelief$ $\textbf{\textit{B}} = (B_1, B_2, \ldots, B_n)$
input 5: $BestPracticeGuidelines$ $\textbf{\textit{G}}$, input 6,7: Short Timer $\textbf{\textit{T}}_{\textbf{\textit{short}}}$, Long Timer $\textbf{\textit{T}}_{\textbf{\textit{long}}}$
$UPDATE()$;
if $S \neq B$
    $Send$ 1st Divergence Alert
        $DeviationCounter++$
    Wait for $\textbf{\textit{T}}_{\textbf{\textit{short}}}$
    $PrevS = S$; $PrevB = B$
    $UPDATE()$;
    if $(PrevS == S$ && $PrevB == B)$
        $Send$ 2st Divergence Alert
        $DeviationCounter++$
    else $Gotoline9$;
    end if;
    Wait for $\textbf{\textit{T}}_{\textbf{\textit{long}}}$
    $PrevS = S$; $PrevB = B$
    $UPDATE()$;
    if $(PrevS == S$ && $PrevB == B)$
        $Send$ Final Divergence
        $DeviationCounter++$ Alert
    else $Gotoline9$;
    end if;
else $DeviationCounter = 0$
    Display $G\prime \subseteq G$ for the current patient state
    $\textbf{\textit{S}} = (S_1, S_2, \ldots, S_n)$
end if
$UPDATE()$ {
    for $i = 1$ $to$ $n$
        $S_i = CurrentState(OA_i)$
        $B_i = CurrentState(PA_i)$
    end for
    $S = (S_1, S_2, \ldots, S_n)$
    $B = (B_1, B_2, \ldots, B_n)$ }
```

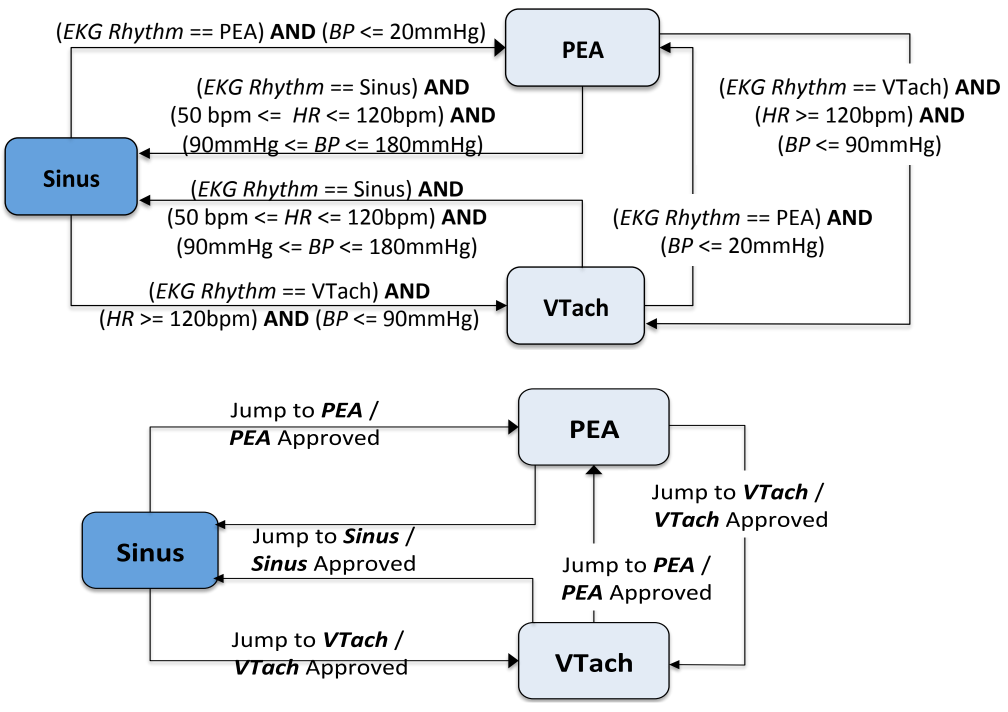

(EKG Rhythm == PEA) AND (BP <= 20mmHg)
PEA
(EKG Rhythm == Sinus) AND (50 bpm <= HR <= 120bpm) AND (90mmHg <= BP <= 180mmHg)
(EKG Rhythm == VTach) AND (HR >= 120bpm) AND (BP <= 90mmHg)
Sinus
(EKG Rhythm == Sinus) AND (50 bpm <= HR <= 120bpm) AND (90mmHg <= BP <= 180mmHg)
(EKG Rhythm == PEA) AND (BP <= 20mmHg)
(EKG Rhythm == VTach) AND (HR >= 120bpm) AND (BP <= 90mmHg)
VTach
Jump to PEA / PEA Approved
PEA
Jump to VTach / VTach Approved
Sinus
Jump to Sinus / Sinus Approved
Jump to PEA / PEA Approved
Jump to VTach / VTach Approved
VTach

Figure 2.7: Arrhythmia State Machine and its Corresponding Physician Belief Machine

2.4.1 Physician Model

At times patient states presented by our system may differ from physician understanding and they may deviate from the best practice guidelines suggested by our system. A source of this inconsistency, should it occur, is that there may be information used by the physician to make a determination but not available to our system, or the inconsistency can be the result of unintended deviation. The physician my override the best practice guidelines at anytime. However, any deviations from the best practice guidelines should be tracked and recorded for offline analysis.

Physician Belief State Machine: We propose a compositional model to handle and track the physician interaction with the system. For each patient state suggested by our model, the response from the physician regarding the new patient state is received and processed by a physician belief state machine. For each pathophysiological state machine, we develop a physician belief state machine with exactly same pathophysiological states. However, the transitions between the states are enabled according to physician response.

Therefore, the two pathophysiological state machines may have different active states at runtime, which indicates physician's intended or unintended deviation. If the physician does not approve the pathophysiological state suggested by the model, the physician state machine does not transition to the suggested state. Moreover, at each time physician may decide to transition (*jump*) to a new state independent of the states suggested by our models. Figure 2.7 shows the arrhythmia state machine and its corresponding physician belief machine.

This compositional approach results in more modular model, in which the organ and physician models can be validated and verified independently. As was shown in Figure 2.5, there is no interaction between a pathophysiological state machine and its corresponding physician machine. Furthermore, physician belief state machines for different pathophysiological processes do not interact either.

We have also added a logging interface to our model that can be used for keeping record of deviations, when the models are integrated into the guidance system. Such records may be used for additional review (can be specifically useful for residents). According to [61], *failure to debrief after code* is one of common errors made in cardiac arrest care and the logging feature can help alleviate this situation. Furthermore, the logs can also be analyzed by the developers to improve the precision of the proposed models.

Convergence-Divergence Protocol: The compositional model allows us to track and record the physician belief on the current states. To allow some degree of flexibility, while reducing the unintended deviation from best practice guidelines we propose the *divergence-convergence* protocol. When there is a discrepancy between a pathophysiological state machine and its corresponding physician state machine an initial alert is generated to reduce any unintended deviation. If patient state continues to deteriorate or remains the same, two additional alerts are generated by the system. Note that all the timer-related values are configurable. After discussion with our physician collaborators, we designed the last alert to be triggered by a longer timer. When we transition to a new pathophysiological state and the best practice and physician diagnoses are still inconsistent, the timer routine restarts. However, if the patient state improves the model (by design) converges with the physician's belief The protocol steps are presented in Pseudocode 1.

2.4.2 Guideline Model

The guideline model encodes best practice monitoring and treatment guidelines. As mentioned previously, physicians often specify these guidelines for different pathophysiological and disease states. Therefore, our proposed modeling paradigm can efficiently encode them.

In most of current clinical guideline models and system, different elements of the patient state information are often used to express preconditions, therapy goals, etc. At each point in time, our model represent the patient state as a set of pathophysiological states (and an aggregated disease state). This representation resemble a patient scenario that can be used to determine the entry point to the guidelines. For example, when patient enters acute lung injury state, the ventilation therapy workflow must be executed. However, different patient state items may also be used to check preconditions or goals at different states of the chosen workflow. For example patient's blood pressure may be monitored to evaluate the effectiveness of fluid resuscitation.

Alarm Management: In this section we also focus on addressing the challenge of alarm management. Our main goal is to reduce slips and lapse caused by inadequate alarm configuration and practices. Such an alarm management must be incorporated into best practice monitoring guidelines for the disease under study.

Most current medical systems raise alerts based on single measurements, which creates considerable false positive alerts. Identifying the relation between measurements and structuring them according to organ pathophysiology adds a layer of intelligence to patient state representation.

We use the notion of ***actionable alerts***, which follows the same line of reasoning as *actionable states*, which was describe in Section 2.3. In fact, by encoding the alerts on patient state changes in already defines actionable states we ensure that they are comprehensive and integrated with pathophysiological sates. These alerts may also be issued on patient's response to treatments or best practice treatment option for the current pathophysiological states.

The alerts on development or resolution of organ insufficiency states are encoded in individual pathophysiological states. These alerts only depend on the information from the corresponding pathophysiological process. There are no interaction between different pathophysiological processes and therefore these alerts can be encoded independent of each other. For example, the alert indicating patient has developed moderate renal insufficiency is encoded in the renal state machine.

However, the alerts on progress of the disease under study is encode in the disease state machine. These alerts require information from several pathophysiological processes. This aggregated information is encoded in the disease states and the disease state machine is the best place to encode this category of alerts. For example, the alert indicating a septic patient has deteriorated to septic shock is encoded in the sepsis state machine.

Best practice guidelines regarding best practice-recommended monitoring extent and frequency may also be encoded similarly. In states with higher degree of insufficiency, the medical staff may require to monitor more frequently or monitor more relevant measurements. Since these monitoring requirements are often categorized according to organ insufficiency and disease progress stage, the corresponding guideline can be encoded in the corresponding pathophysiological and disease states. Each alert message has the following items:

- The new disease/pathophysiological state (and the corresponding process).
- Patient state change type: improvement or deterioration.
- The current value of relevant measurements.
- The normal range for each of the above measurements.
- Recommended treatment and monitoring actions.

Note that the treatment actions recommended in these alert messages are presented in abstract; more detailed treatment alerts will be discussed later. Examples of alert messages will be presented in Chapters 3 and 4.

2.5 Model Verification

As mentioned in Chapter 1, the desired model behaviors must always hold, not just in the limited clinical simulation. We address this challenge by performing model checking and formally verifying the safety and system properties.

To formally verify the correctness of the models, we use UPPAAL model checker tool. Currently, the translation from Yakindu to UPPAAL is done manually. The correctness of the translation is a challenge that also needs to be addressed but is outside the scope of this paper. However, the UPPAAL model-checker is also based on the theory of timed automata and this make the translation process less complex. In [62] automatic translation of Yakindu models to UPPAAL is discussed. However, in this paper the translation is done manually until such research works mature.

As mentioned before, we use an open source tool, called Yakindu, to develop the models. The tool provides an integrated modeling environment and development based on the concept of statecharts. To make the translation to UPPAAL less complex, we avoided using composite states, supported by statecharts, in the Yakindu models. UPPAAL model checker tool does not support composite states. Therefore, using them in our organ models would make the task of translation to UPPAAL more complex and possibly error-prone. Another reason for avoiding composite states is to maintain the model simplicity for clinical validation by physicians. Understanding complex computer science concepts, such as composite states, would be a non-trivial tasks for the physicians. Our goal is to keeps the models as simple and intuitive as possible. Note that we use the orthogonal regions supported in statecharts to represent a set of concurrent organ models, which we develop to track concurrent changes in the patient state. In UPPAAL, a system is also modeled as a network of communicating state machines. This makes the task of translation to UPPAAL and formal verification more straightforward.

The Yakindu models may include state and transition actions. Since UPPAAL only supports transition actions, we also translated any necessary state action to a transition action. In Yakindu, the state actions include entry, exit and timer based actions. The entry action should be performed once when entering the state. We translated this to an action assigned to all the incoming transitions to the corresponding state. The exit action, which must be executed when existing the state, was translated to actions on all the outgoing transitions. For the timer-based state actions, we created a self-transition and used the timer as transition guard. Figure 3.6 shows the Yakindu *arrhythmia* model and its corresponding UPPAAL model, constructed manually. The four states are mapped to four locations, each representing an arrhythmia state. The entry state actions are translated to updates on all the incoming transitions to the corresponding states.

As explained previously, and also illustrated in Figure 2.5, there is no direct interaction among dif-

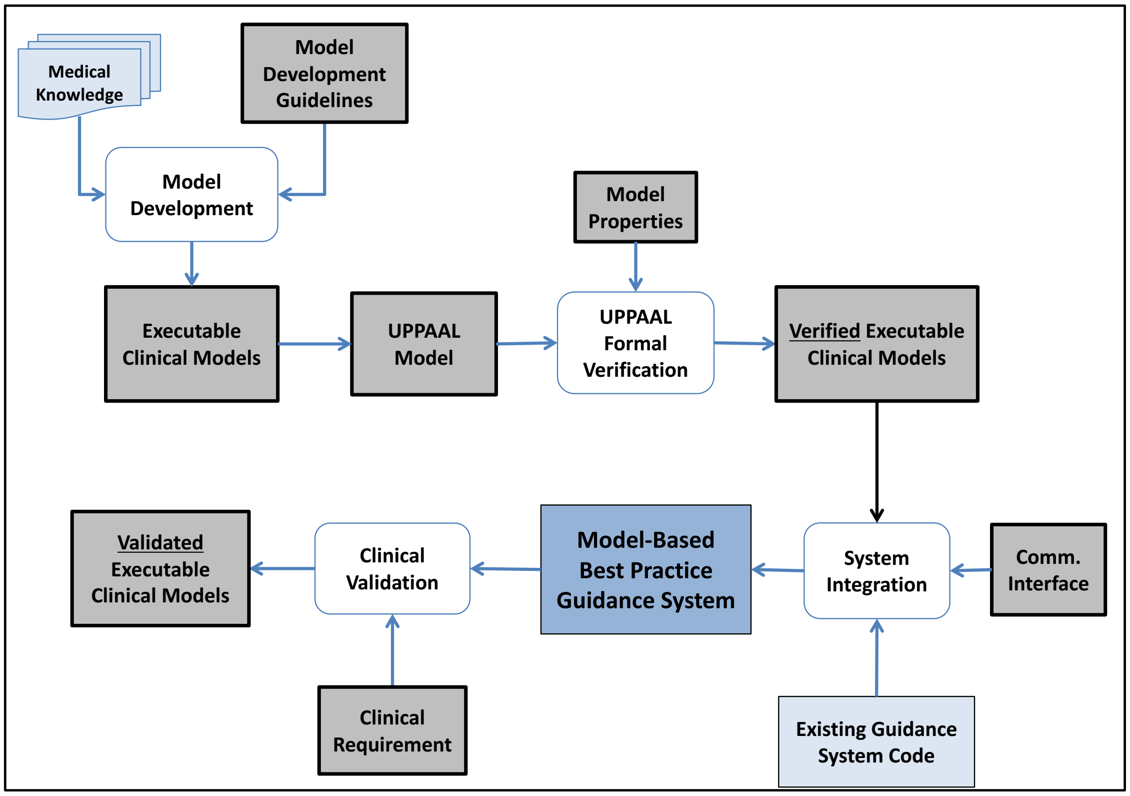

Medical Knowledge
Model Development Guidelines
Model Development
Model Properties
Executable Clinical Models
UPPAAL Model
UPPAAL Formal Verification
Verified Executable Clinical Models
Validated Executable Clinical Models
Clinical Validation
Model-Based Best Practice Guidance System
System Integration
Comm. Interface
Clinical Requirement
Existing Guidance System Code

Figure 2.8: Model-Driven Design for Clinical Guidance System

ferent pathophysiological state machines or physician belief state machines. Furthermore, a pathophysiological state machine does not interact with its corresponding physician belief state machine either. This improves verification modularity and therefore we can verify most of the developed models separately. All the interaction are encapsulated in the disease and guideline models and their correctness and safety properties must be verified as part of verification of these modeling components.

After constructing the UPPAAL models, (a subset of identified) clinical requirements must be formalized as (timed) computation tree logic, CTL, formulas and verified using UPPAAL model checker. The exact formula depends on the disease under study. We present the set of verified propertied for our cardiac arrest model in Chapter 3.

2.6 Model Integration

As discussed earlier, developing efficient and accurate clinical models is the first step in design of guidance systems. As the next step, the models must be integrated into the guidance to derive

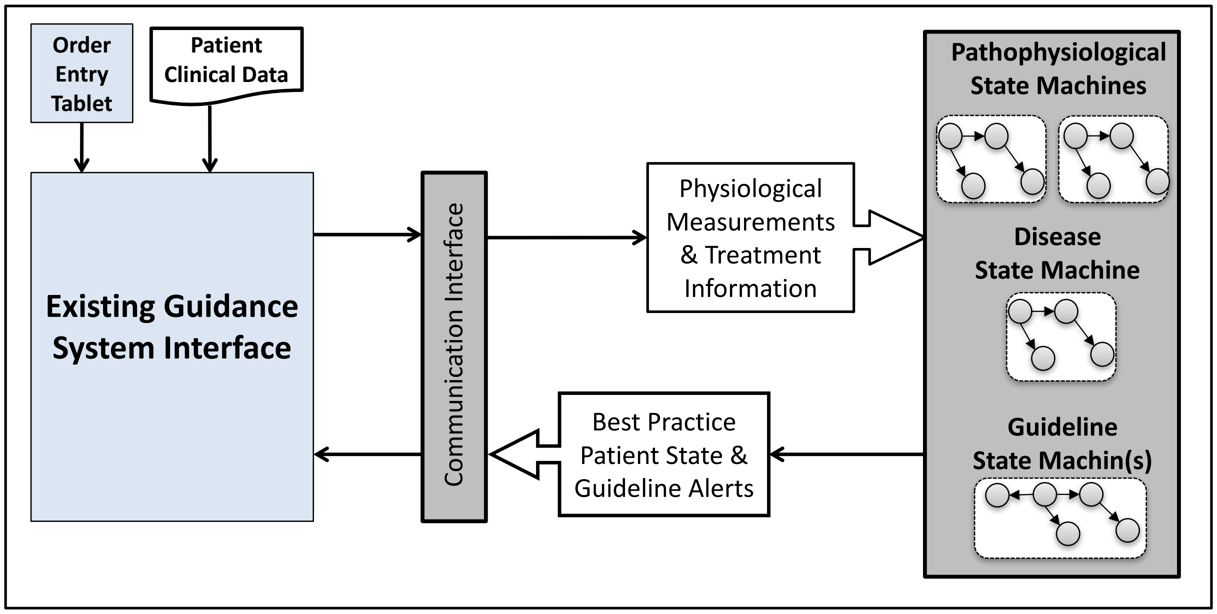

Order Entry Tablet
Patient Clinical Data
Existing Guidance System Interface
Communication Interface
Physiological Measurements & Treatment Information
Best Practice Patient State & Guideline Alerts
Pathophysiological State Machines
Disease State Machine
Guideline State Machin(s)

Figure 2.9: System Overview

its execution. The models interact with the guidance system using our handwritten communication code.The model integration also provides a more user-friendly environment for simulation of different clinical scenarios. This would allow physicians to validate the correctness and usefulness of developed models more efficiently.

Figure 2.8 shows the approach we have followed to integrate the developed models with an existing guidance system. It also summarizes the steps involved in the construction, verification and integration of the clinical models. At the first step, we translated the generic medical knowledge into a set of communicating finite state machines. We used our proposed organ-centric paradigm, in which we model patient state as a network of pathophysiological and disease state machines. We also used the finite state machine formalism to develop a best practice guideline model that encodes all the monitoring and treatment guidelines. A compositional approach was also proposed to handle and track the physician interaction with the system. We use the open source tool, Yakindu, to develop the models. The tool provides an integrated modeling environment and development based on the concept of statecharts. The tool also allows simulation of the developed model to validate its dynamic behavior [8]. Next, the finalized Yakindu models were manually translated to UPPAAL for exhaustive verification of clinical and system requirements. The clinical requirements are formalized as computation tree logic formulas and along side the system requirements are verified using UPPAAL. Note that any necessary modification to UPPAAL model are reflected back to the Yakindu Model.

Now, at this step we integrate the verified models with an existing guidance system interface, using some handwritten communication code. The Java code can be generated automatically from the Yakindu model and integrated with the existing system using some hand-written glue code. However, the tool also allows direct interaction between the Yakindu model and existing Java code. Therefore, for visualization purposes we have integrated our cardiac arrest and sepsis model with an existing guidance systems, previously developed by our research group. The models interact with the existing system using our handwritten communication code. The patient data entered into the guidance system is communicated to our models through the communication (glue) code. This allows use simulate different clinical scenarios. The generated monitoring and treatment guidelines by the models are sent to the guidance system to be displayed for medical staff. The physician response to the generated guidelines are also recorded by the system and communicated back to our model [3].Figure 2.9 shows the communication flow between the models and guidance system.

2.7 Clinical Validation

Through simulation of our model-driven guidance system, we validate clinical correctness and usefulness of the models. A set of clinical requirements have been drafted in collaboration with physicians. In the process of developing our organ-centric pathophysiological organ models, we have gone through multiple iterations of design, validation and close discussion with our ICU physician collaborators. The executable organ models allow us to simulate the disease under study. The clinical simulation provides an environment for dynamic validation of the models. Physicians can also directly look at these organ models and cross reference with the specified clinical requirements. Note that, the developed models are at same level of abstraction the medical knowledge and reasoning are known to physician. At our proposed level of abstraction, physicians with minimal IT background can easily understand the models and review the design. If directly implemented in procedural style using a high level language such as Java, organ-centric patient state representation could not be easily validated by physicians. We

[3]In our sepsis guidance system, we have added Approve and Hold buttons to our guideline alert messages. The physician response is recorded by tracking which button is clicked for each guideline item.

then apply the necessary changes, as suggested by physician(s), and re-validate the refined model. The following are a subset of clinical requirements validated for the developed organ models:

- CR_1: All the pathophysiological processes relevant to the disease under study are modeled.
- CR_2: All the insufficiency states, recognized by medical textbooks, are modeled for each pathophysiological process.
- CR_3: The correct and only necessary subset of physiological measurements are used to define the states in each automaton.
- CR_4: The definition of insufficiency states are correct.
- CR_5: Any changes in the physiological measurements will result in transition(s) to the correct insufficiency state(s).

We have shown our developed models and simulation of cardiac arrest scenario to our physician collaborators for clinical validation. More details are presented in the next chapter. Note that each pathophysiological process and its corresponding physician belief state machine has the exact same states. The transitions are simply enabled by the physician input. The correctness of the physician models are formally verified using UPPAAL.

Next, we describe how the modeling guidelines, presented in Section 2.3, can be applied to construct the cardiac arrest and sepsis models. We provide more details on formal verification, integration and validation of the constructed models. In Chapter 3, we also discuss the preliminary clinical validation we have conducted of our model-driven guidance system at a Carle ACLS training class.

2.8 Research Publications

The following is our published research paper on organ-centric modeling paradigm and integration of the models into a best practice guidance system:

- **M. Rahmaniheris**, P. Wu, L. Sha, and R. B., Berlin, ”*An Organ-Centric Best Practice Assist System for Acute Care*”, IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS 2016) [63]

We are in the process of preparing the following paper, on the proposed model-driven guidance system design and evaluation. This paper will be submitted to IEEE Real-Time Systems Symposium (RTSS) 2017:

- **M. Rahmaniheris**, Y. Jiang, P. Wu, and L. Sha, *Design and Evaluation of Model-Driven Guidance Systems for Acute Care*"

Chapter 3

Cardiac Arrest Case Study

We first briefly describe the cardiovascular organ system and its pathophysiology. This is by no mean a thorough description. The cardiovascular system consists of the heart and blood vessel. The main responsibility of this organ system is to support blood circulation to provide nutrients, oxygen, etc. to different parts of the human body. Another task performed by this organ system is to support the $CO2$ and $O2$ exchanges in the lungs that stabilize blood PH, which is the measure of acidity of the blood. We can categorize the different pathophysiological states of the cardiovascular organ system into four main insufficiencies. The first category represent the conditions, in which the blood flow is blocked or partially blocked. This can be the result of different cardiovascular disease such as blood clot, hardening and narrowing of the artery, peripheral artery disorder or circulatory shock. The second category represents the abnormalities in heart rhythm, which leads to significant drop of cardiac output and impairs blood circulation. The insufficient blood circulation in the lungs to perform the $CO2$ and $O2$ exchange results in $CO2$ accumulation in the blood and causes a rise in blood PH level. This plays an important role in our case study on cardiac arrest. Abnormal blood gas levels and dysregulated blood pressure are the third and fourth categories. Table 5.1 shows a subset of cardiovascular states relevant to the intelligent management of devices. We have verified this abstract model of cardiovascular insufficiency states with the physician, who collaborates closely with our research group.

Cardiac arrest is the sudden loss of heart function, which is caused by the loss of the heart's electrical and muscular pumping function. Following cardiac resuscitation efforts, the heart rhythm may change and include abnormal or irregular heart rhythms or waveforms (arrhythmias). The heart may beat chaotically during resuscitation, which leads to significant drop of cardiac output and impairs blood circulation. The insufficient blood circulation in the lungs impedes the CO_2 and O_2 exchange resulting in CO_2 accumulation in the blood, which may be related to a decrease in blood pH level.

Cardiac arrest can lead to death within minutes. The guideline provided by American Heart Association (*AHA*) to reverse the cardiac arrest includes cardiopulmonary resuscitation (*CPR*), using the defibrillator device to shock the heart by delivering a therapeutic dose of electrical energy to the heart, and administration of intravenous drugs such as epinephrine [64], [65]. Epinephrine is a vessel constrictor that stimulates the heart and increases the arterial blood pressure.

The patient's blood pressure, heart rate, blood O_2 saturation must be monitored continuously. An electrocardiography machine, EKG machine, must be attached to the patient to monitor the heart's electrical activity and rhythm. However, the defibrillator may only be used if a shockable rhythm is present. *Pulseless ventricular tachycardia* and *ventricular fibrillation* are considered shockable. If the EKG machine indicates *pulseless electrical activity* or *asystole*, which is flat line, shocking a patient's heart will be ineffective and is commonly contraindicated. If the rhythm is not shockable, epinephrine is often administered through intravenous access.

A patient in cardiac arrest may develop acidosis which is indicated when blood gas analysis is performed during resuscitation. In this cardiovascular state blood acidity level is abnormally high, indicated by low value of blood pH. This is caused by poor tissue perfusion and blood oxygenation during cardiac arrest. On the other hand, acidosis if severe may be one of the potential adverse conditions under which epinephrine may be less effective. At times, it is recommended to correct the severe acidosis by improving the CO_2 and O_2 exchange and by administering to the patient a base medication, sodium bicarbonate. There are times when such therapy may improve the effectiveness of epinephrine.

In addition to the cardiovascular organ system, as resuscitation efforts continue, it may be found that the lung (pulmonary) and kidney (renal) organ systems are also affected. In this paper we demonstrate a subset of possible interactions between the cardiac and kidney organ systems in our computational model. Moderate or severe renal insufficiency can affect the treatment plan for a the cardiac arrest patient. For a patient with kidney insufficiency, there may be times when other medications may be preferred in place of epinephrine due to some reports of epinephrine toxicity in the face of kidney dys-

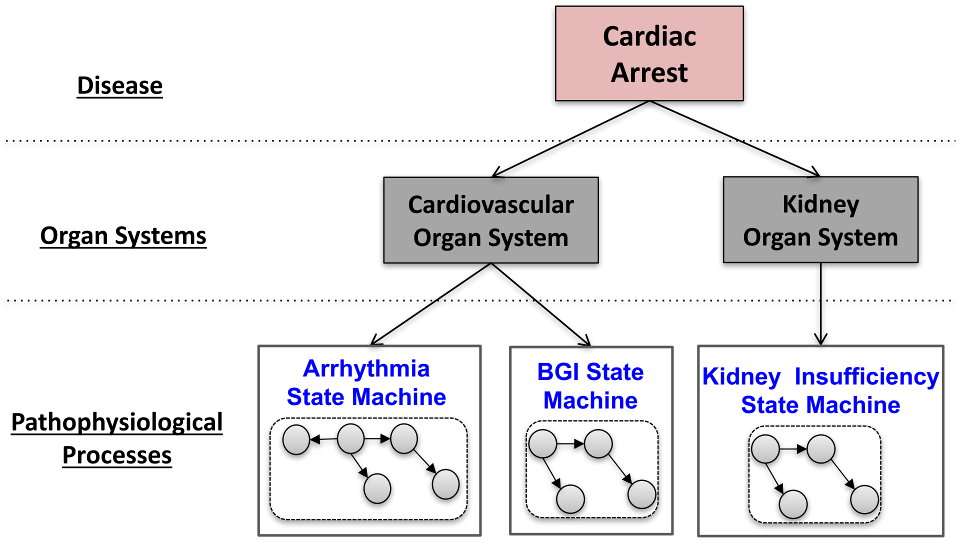


Figure 3.1: The Relevant Pathophysiological Processes for Cardiac Arrest

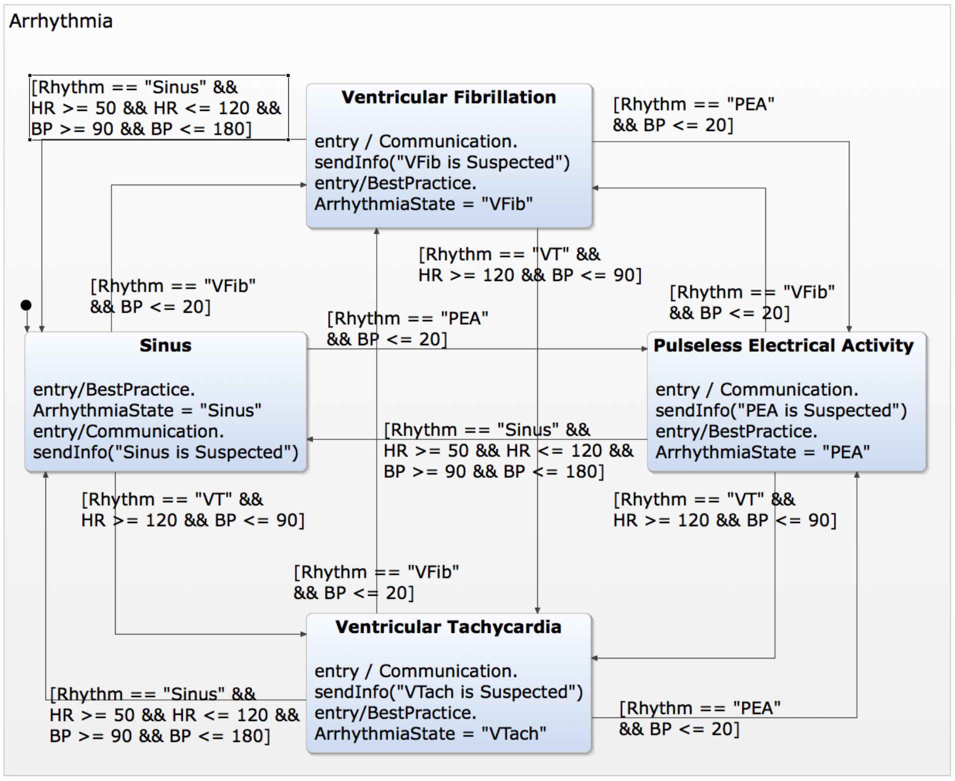
function. Other vasoactive medications to consider may include dobutamine and levophed depending on patient condition [66]. In case of severe renal insufficiency, where the patient shows signs of hyperkalemia [1], it may also be recommended to use calcium chloride infusion through the intravenous to treat the effects of a very high potassium level on the heart.

In the following sections we discuss how the *organ-centric pathophysiology-based cardiac arrest model* can be constructed by applying the modeling guidelines presented in Section 2.3.

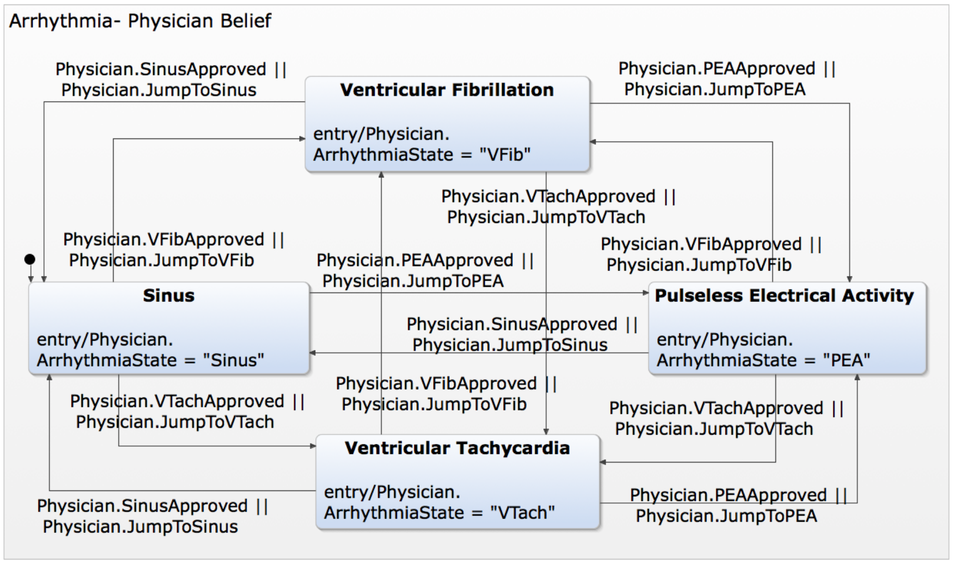
3.1 Model Construction

We start by applying our first modeling guideline, $\mathbf{G_1}$. Cardiac arrest causes several abnormalities mainly in cardiovascular organ system. These include abnormal heart rhythm and imbalance in arterial blood gas levels, including blood acidity level. According to this guideline, a pathophysiological process becomes relevant if the corresponding physiological function is affected by the disease or affect the treatment. Therefore, blood gas imbalance and arrhythmia are relevant pathophysiological processes that must be tracked when monitoring and treating a cardiac arrest patient. In addition to the cardiovascular organ system, as describe above, it may be found that the kidney (renal) organ system is also affected. On the other hand, moderate or severe renal insufficiency can affect the treatment can *affect*

[1]High level of potassium in blood

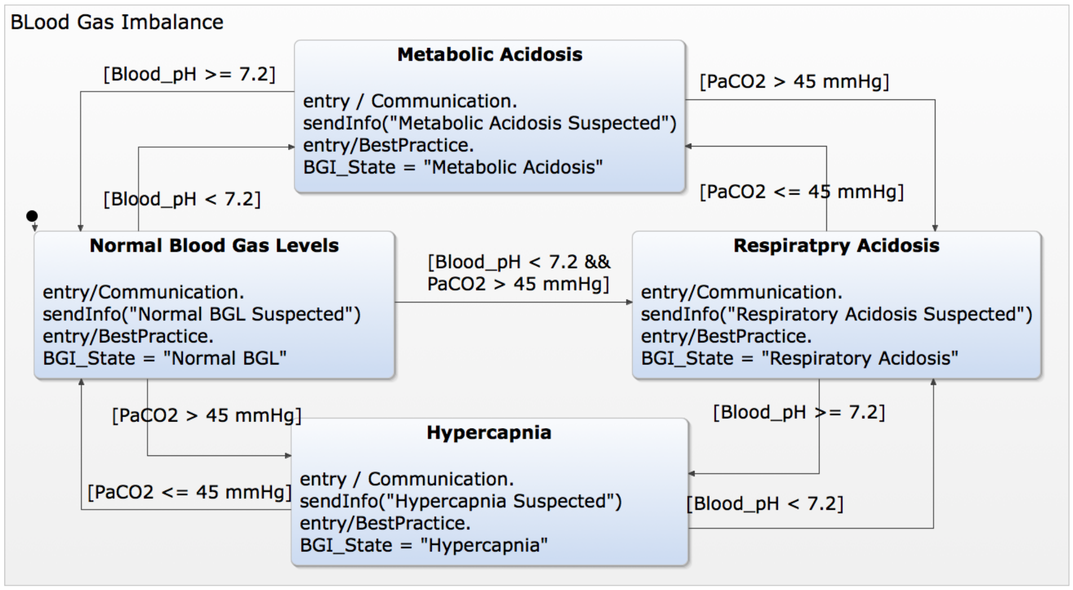

Arrhythmia
[Rhythm == "Sinus" && HR >= 50 && HR <= 120 && BP >= 90 && BP <= 180]
Ventricular Fibrillation
entry / Communication. sendInfo("VFib is Suspected")
entry/BestPractice. ArrhythmiaState = "VFib"
[Rhythm == "PEA" && BP <= 20]
[Rhythm == "VT" && HR >= 120 && BP <= 90]
[Rhythm == "VFib" && BP <= 20]
[Rhythm == "VFib" && BP <= 20]
[Rhythm == "PEA" && BP <= 20]
Sinus
entry/BestPractice. ArrhythmiaState = "Sinus"
entry/Communication. sendInfo("Sinus is Suspected")
Pulseless Electrical Activity
entry / Communication. sendInfo("PEA is Suspected")
entry/BestPractice. ArrhythmiaState = "PEA"
[Rhythm == "Sinus" && HR >= 50 && HR <= 120 && BP >= 90 && BP <= 180]
[Rhythm == "VT" && HR >= 120 && BP <= 90]
[Rhythm == "VT" && HR >= 120 && BP <= 90]
[Rhythm == "VFib" && BP <= 20]
Ventricular Tachycardia
entry / Communication. sendInfo("VTach is Suspected")
entry/BestPractice. ArrhythmiaState = "VTach"
[Rhythm == "Sinus" && HR >= 50 && HR <= 120 && BP >= 90 && BP <= 180]
[Rhythm == "PEA" && BP <= 20]

(a) Best Practice

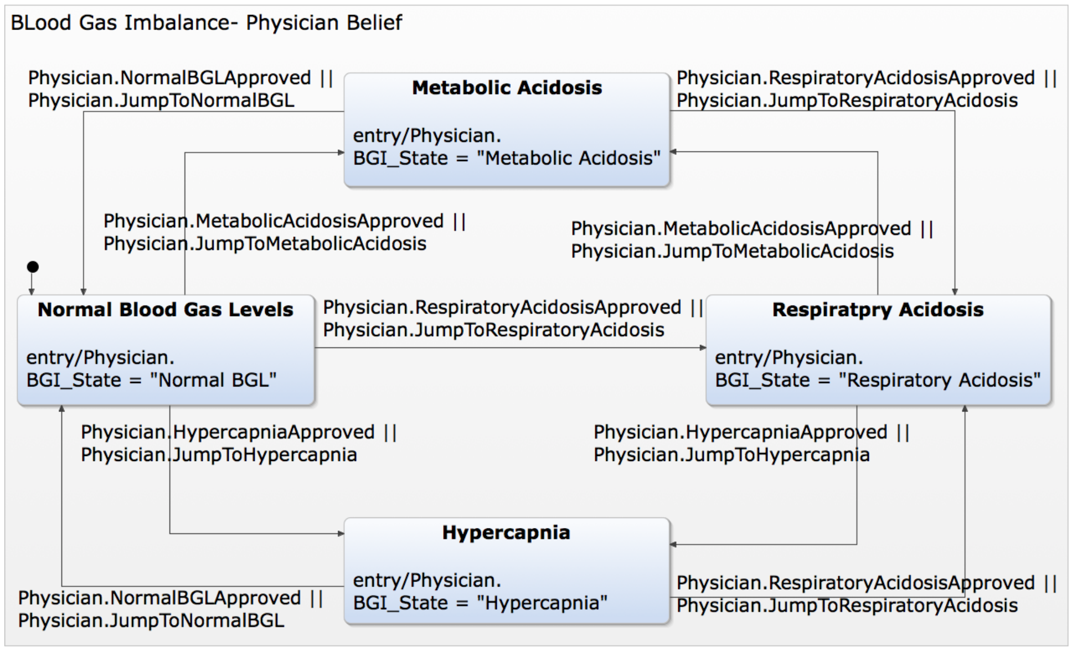

Arrhythmia- Physician Belief
Physician.SinusApproved || Physician.JumpToSinus
Ventricular Fibrillation
entry/Physician. ArrhythmiaState = "VFib"
Physician.PEAApproved || Physician.JumpToPEA
Physician.VTachApproved || Physician.JumpToVTach
Physician.VFibApproved || Physician.JumpToVFib
Physician.PEAApproved || Physician.JumpToPEA
Physician.VFibApproved || Physician.JumpToVFib
Sinus
entry/Physician. ArrhythmiaState = "Sinus"
Pulseless Electrical Activity
entry/Physician. ArrhythmiaState = "PEA"
Physician.SinusApproved || Physician.JumpToSinus
Physician.VFibApproved || Physician.JumpToVFib
Physician.VTachApproved || Physician.JumpToVTach
Physician.VTachApproved || Physician.JumpToVTach
Ventricular Tachycardia
entry/Physician. ArrhythmiaState = "VTach"
Physician.SinusApproved || Physician.JumpToSinus
Physician.PEAApproved || Physician.JumpToPEA

(b) Physician Belief

Figure 3.2: Arrhythmia State Machines



(a) Best Practice


BLood Gas Imbalance- Physician Belief
Physician.NormalBGLApproved ||
Physician.JumpToNormalBGL
Metabolic Acidosis
entry/Physician.
BGI_State = "Metabolic Acidosis"
Physician.RespiratoryAcidosisApproved ||
Physician.JumpToRespiratoryAcidosis
Physician.MetabolicAcidosisApproved ||
Physician.JumpToMetabolicAcidosis
Physician.MetabolicAcidosisApproved ||
Physician.JumpToMetabolicAcidosis
Normal Blood Gas Levels
entry/Physician.
BGI_State = "Normal BGL"
Physician.RespiratoryAcidosisApproved ||
Physician.JumpToRespiratoryAcidosis
Respiratpry Acidosis
entry/Physician.
BGI_State = "Respiratory Acidosis"
Physician.HypercapniaApproved ||
Physician.JumpToHypercapnia
Physician.HypercapniaApproved ||
Physician.JumpToHypercapnia
Hypercapnia
entry/Physician.
BGI_State = "Hypercapnia"
Physician.NormalBGLApproved ||
Physician.JumpToNormalBGL
Physician.RespiratoryAcidosisApproved ||
Physician.JumpToRespiratoryAcidosis

(b) Physician Belief

Figure 3.3: Blood Gas Imbalance State Machines

the choice of vasoactive agent to reverse cardiac arrest. Therefore, renal insufficiency process becomes relevant to cardiac arrest, because not only the kidney organ system may be affected by the disease but also the renal insufficiency state can affect its treatment plan. We also had a series of discussions with our ICU physician collaboratives to confirm the identified pathophysiological processes for our cardiac arrest model.

Next, we follow $\mathbf{G_2}$ and identify the information that is important, but not need to be encoded in the same details as others in our cardiac arrest model. As explained above, kidney is not the main affected organ system and only the severity of renal insufficiency, if any, must be known to safely and effectively adapt the treatment plan. Therefore, the medical information regarding the pathophysiology of kidney organ system are aggregated as degree of renal insufficiency. On the other hand, more derails are needed regarding the cardiovascular system. For example, we encode arrhythmia and blood gas imbalance separately in our cardiac arrest model. It is not simply sufficient to represent all the changes in the cardiovascular organ system by an aggregated degree of insufficiency.

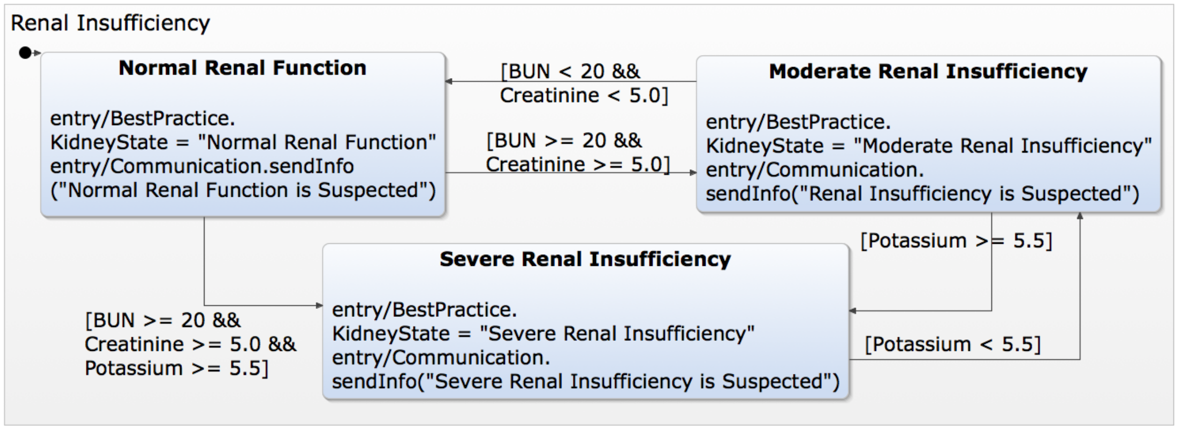
At this stage, we must determine the concurrency level of our cardiac arrest model. According to $\mathbf{G_3}$, the affected physiological functions that need to be tracked simultaneously must be modeled by separate finite state machines. If $\mathbf{G_2}$ has already been applied, following this guideline becomes straightforward. Previously, we specified the different (aggregated) pathophysiological states need to be tracked for a cardiac arrest patient. To keep track of all these states and prevent any critical changes from being overlooked we model each process using a separate state machine. Therefore, we need to develop three state machines for the cardiac arrest model: *arrhythmia*,*blood gas imbalance (BGI)*, and *renal insufficiency*. Figure 3.1 shows our proposed hierarchical model for the cardiac arrest.

Now, must define the states for each of the identified pathophysiological state machines. There are measurements, which physicians often interpret together to have a better assessment. In the arrhythmia state machine each state is defined by correlating EKG rhythm with blood pressure and heart rate. To assess the degree of renal insufficiency creatinine, BUN and potassium are looked at as a group. We use these three measurement as the state variables for the renal insufficiency state machine. The measurements in *ABG* test, $PaO2$, $PaCO2$ and pH, are used to define different blood gas imbalance states.

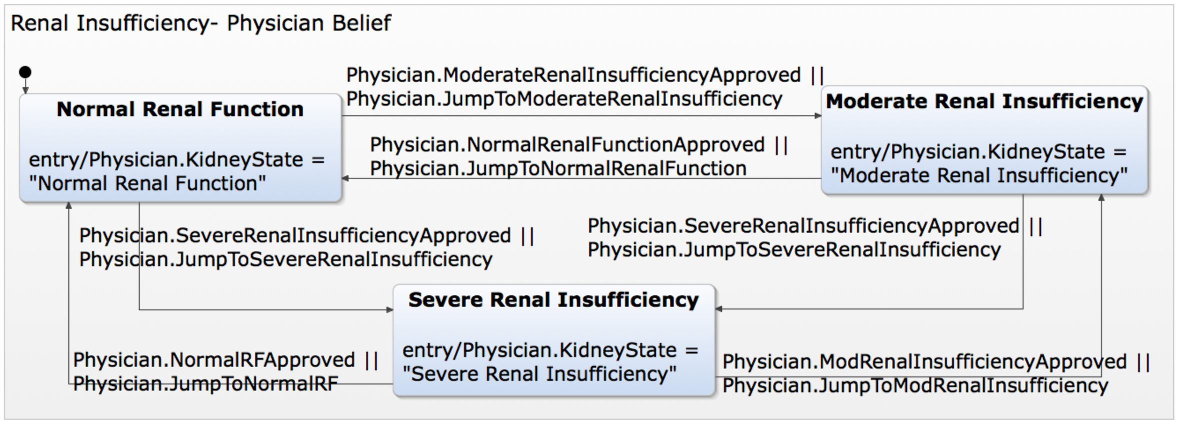
However, defining a large number of states can make the model very complex and unmanageable. In guideline $\mathbf{G_4}$, we introduced the notions of *actionable states*. Not all the changes in physiological measurements are relevant or useful for a given disease. Providing too much information and too many alerts to the medical staff may create confusion and increase their mental overload. We define a new state when there is a need for new monitoring and therapeutic plan. The actionable states can be extracted from the requirements that specify the organ states, which trigger an action from the medical staff. For example, to adapt the cardiac arrest treatment plan, we only require to know if the patient has developed any renal insufficiency, and whether it is moderate or severe. Therefore, we can identify three actionable states for the kidney state machine: **Normal Renal Function**, *Moderate Renal Insufficiency* and *Severe Renal Insufficiency*. Fo the other tw state machines the actionable alerts can be directly extracted for best practice guidelines. In ACLS algorithm, which is the workflow provided by American Heart Association to reveres cardiac arrest, four arrhythmia states are identified: $Ventricular\ Tachycardia$, $Ventricular\ Fibrillation$, $Asystole$ and PEA. Therefore, we define five actionable states, including these four arrhythmia states and an additional state representing sinus rhythm. However, we decided not to define a separate state for *Asystole*, since it can be easily detected and has the same treatment plan as PEA. Lastly, we have identified three main arterial blood gas abnormalities from medical literature: *metabolic acidosis*, *hypercapnia*, and *respiratory acidosis*. We define three actionable states corresponding to these identified abnormalities and an additional state to represent normal blood gas levels.

After defining the states, we must identify the correct and optimal number of transitions by considering the underlying medical knowledge. According to guideline $\mathbf{G_5}$, when continuous physiological variables are used to define different states, the transitions are only defined to represent the progress of insufficiency. This case applies to *blood gas imbalance (BGI)*, and *renal insufficiency* state machines. We only define an edge between the states that represent gradual deterioration and improvement. Note that, since we have only three actionable states in the renal state machine and we define a transition from the initial state to all the others, the application of this guidelines cannot be explicitly observed from the final constructed model. In the *arrhythmia* state machine, we require to define transition between all insufficiency states since a discrete measurement, that is KG rhythm, is used as state variable. Figures 3.2(a)), 3.3(a), and 3.4(a)) show the final constructed state machines resulted from application of the modeling guidelines. Figures 3.2(b)), 3.3(b), and 3.4(b)) show the corresponding physician models for each pathophysiological process. They are constructed following the approach described in Section 2.4.1. All the models have been developed using Yakindu.

Note that the last guideline, $\mathbf{G_6}$, has been partially followed since we do not define any interaction


Renal Insufficiency
Normal Renal Function
entry/BestPractice. KidneyState = "Normal Renal Function" entry/Communication.sendInfo ("Normal Renal Function is Suspected")
[BUN < 20 && Creatinine < 5.0]
[BUN >= 20 && Creatinine >= 5.0]
Moderate Renal Insufficiency
entry/BestPractice. KidneyState = "Moderate Renal Insufficiency" entry/Communication. sendInfo("Renal Insufficiency is Suspected")
[Potassium >= 5.5]
Severe Renal Insufficiency
entry/BestPractice. KidneyState = "Severe Renal Insufficiency" entry/Communication. sendInfo("Severe Renal Insufficiency is Suspected")
[BUN >= 20 && Creatinine >= 5.0 && Potassium >= 5.5]
[Potassium < 5.5]

(a) Best Practice


Renal Insufficiency- Physician Belief
Physician.ModerateRenalInsufficiencyApproved || Physician.JumpToModerateRenalInsufficiency
Normal Renal Function
entry/Physician.KidneyState = "Normal Renal Function"
Physician.NormalRenalFunctionApproved || Physician.JumpToNormalRenalFunction
Moderate Renal Insufficiency
entry/Physician.KidneyState = "Moderate Renal Insufficiency"
Physician.SevereRenalInsufficiencyApproved || Physician.JumpToSevereRenalInsufficiency
Physician.SevereRenalInsufficiencyApproved || Physician.JumpToSevereRenalInsufficiency
Severe Renal Insufficiency
entry/Physician.KidneyState = "Severe Renal Insufficiency"
Physician.NormalRFApproved || Physician.JumpToNormalRF
Physician.ModRenalInsufficiencyApproved || Physician.JumpToModRenalInsufficiency

(b) Physician Belief

Figure 3.4: Renal Insufficiency State Machines

between the three pathophysiological state machines. We codify any dependency and indirect interaction between them in a central guideline model.

3.1.1 Cardiac Arrest Guideline Model

Thee guideline model for cardiac arrest has been also developed using finite state machine formalism. The monitoring and treatment guidelines are categorized according to patient pathophysiological states. We define the patient state S as a tuple $(S_1, S_2, \ldots, S_n)$, in which S_i is the current state in ith pathophysiological process and n is the total number of these processes developed for the disease under study. In cardiac arrest model, $n = 3$ and S = $(VTach$, $Metabolic\ Acidosis$, $Moderate\ Renal$

$Insufficiency$) can be an example of patient aggregated state.

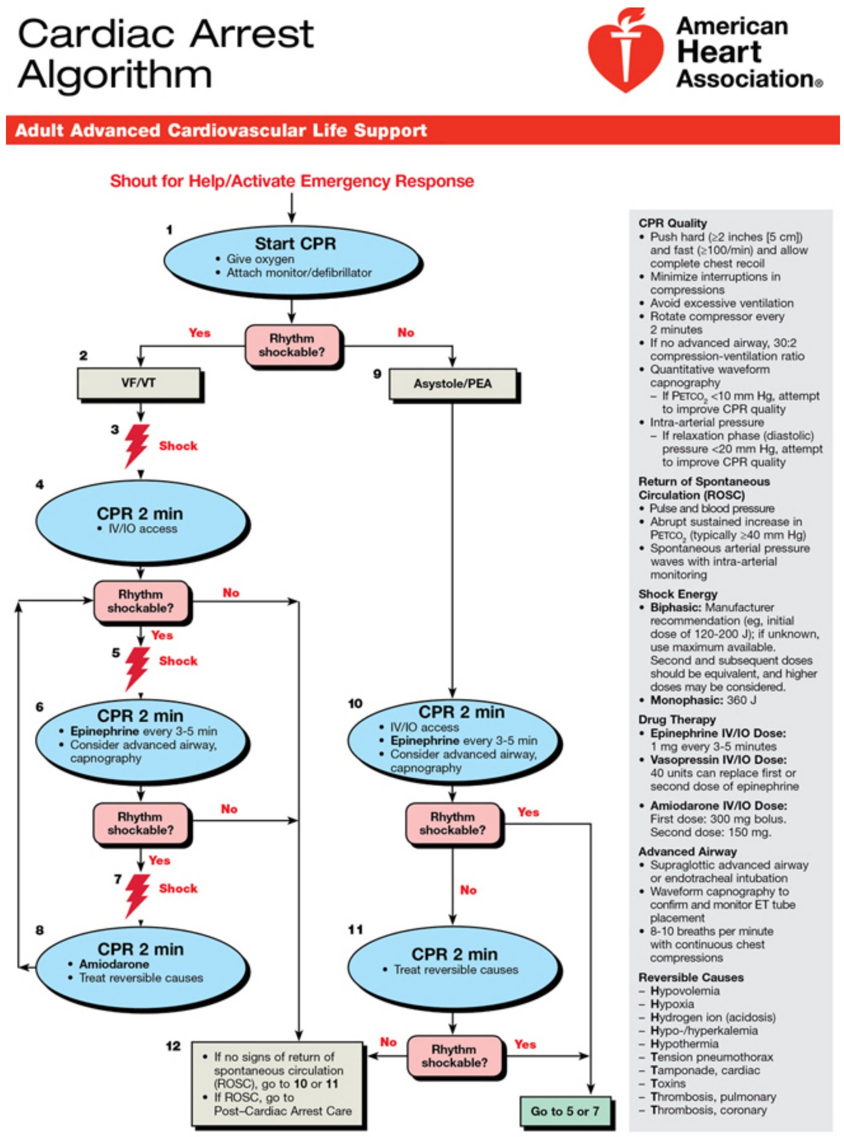
The guideline model for the cardiac arrest scenario codifies the ACLS[2] algorithm provided by American Heart Association (*AHA*) and other well-established guidelines recommended by our physician collaborators. We model the cardiac arrest guidelines using a single state machine. Note that, the guideline model may also include several state machines, which can be executed concurrently or their execution can be synchronized using another top-level guideline manager. We provide more details when discussing sepsis case study in Chapter 4. Figure 3.5 show the ACLS algorithm.

The cardiac arrest guideline state machine periodically communicates with all the organ models to get their current states and determine the current patient state. As mentioned before, the encoded guidelines are categorized to patient state defined as explained above. Therefore, when the patient states changes, the best practice model determines the subset of guidelines for the current patient state. The guideline state machine codifies any potential interaction between different modeling components. In the above example, treatment plan for $VTach$ may need to be adapted because the patient has developed renal insufficiency. The guideline state machine contains all the required information to encode interactions or other dependency relations such as the one just described. The following are examples of encoded cardiac arrest guidelines:

- All medical staff must consider *shock* when $VTach$ or $VFib$ rhythm is confirmed.
- All medical staff must avoid *shock* when PEA or $Asystole$ rhythm is confirmed
- The physician must consider vasoactive agents such as epinephrine to correct arrhythmia.
- *Epinephrine* may not be effective for a cardiac arrest patient with low blood pH level. It may be recommended to consider Bicarbonate to treat the high level of acidity.
- For a cardiac arrest patient with renal insufficiency, it may be recommended to adjust fluid resuscitation parameters and substitute epinephrine with other vasoactive agents such as *levophed*, which are less harmful to kidney.

In Section 3.3, we explain how the constructed pathophysiological and guideline models are used

[2]Advanced Cardiac Life Support



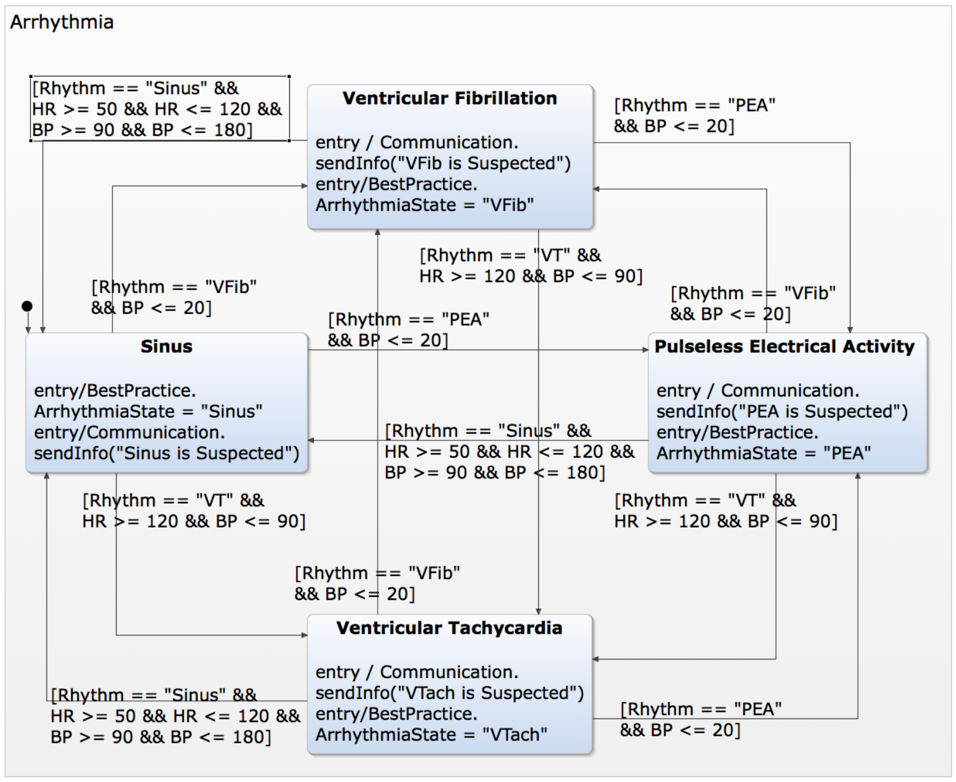
Cardiac Arrest Algorithm
American Heart Association
Adult Advanced Cardiovascular Life Support
Shout for Help/Activate Emergency Response
1
Start CPR
• Give oxygen
• Attach monitor/defibrillator
Rhythm shockable?
Yes
No
2
VF/VT
9
Asystole/PEA
3
Shock
4
CPR 2 min
• IV/IO access
Rhythm shockable?
No
Yes
5
Shock
6
CPR 2 min
• Epinephrine every 3-5 min
• Consider advanced airway, capnography
Rhythm shockable?
No
Yes
7
Shock
8
CPR 2 min
• Amiodarone
• Treat reversible causes
10
CPR 2 min
• IV/IO access
• Epinephrine every 3-5 min
• Consider advanced airway, capnography
Rhythm shockable?
Yes
No
11
CPR 2 min
• Treat reversible causes
Rhythm shockable?
No
Yes
12
• If no signs of return of spontaneous circulation (ROSC), go to 10 or 11
• If ROSC, go to Post–Cardiac Arrest Care
Go to 5 or 7
CPR Quality
• Push hard (≥2 inches [5 cm]) and fast (≥100/min) and allow complete chest recoil
• Minimize interruptions in compressions
• Avoid excessive ventilation
• Rotate compressor every 2 minutes
• If no advanced airway, 30:2 compression-ventilation ratio
• Quantitative waveform capnography
– If PETCO2 <10 mm Hg, attempt to improve CPR quality
• Intra-arterial pressure
– If relaxation phase (diastolic) pressure <20 mm Hg, attempt to improve CPR quality
Return of Spontaneous Circulation (ROSC)
• Pulse and blood pressure
• Abrupt sustained increase in PETCO2 (typically ≥40 mm Hg)
• Spontaneous arterial pressure waves with intra-arterial monitoring
Shock Energy
• Biphasic: Manufacturer recommendation (eg, initial dose of 120-200 J); if unknown, use maximum available. Second and subsequent doses should be equivalent, and higher doses may be considered.
• Monophasic: 360 J
Drug Therapy
• Epinephrine IV/IO Dose: 1 mg every 3-5 minutes
• Vasopressin IV/IO Dose: 40 units can replace first or second dose of epinephrine
• Amiodarone IV/IO Dose: First dose: 300 mg bolus. Second dose: 150 mg.
Advanced Airway
• Supraglottic advanced airway or endotracheal intubation
• Waveform capnography to confirm and monitor ET tube placement
• 8-10 breaths per minute with continuous chest compressions
Reversible Causes
– Hypovolemia
– Hypoxia
– Hydrogen ion (acidosis)
– Hypo-/hyperkalemia
– Hypothermia
– Tension pneumothorax
– Tamponade, cardiac
– Toxins
– Thrombosis, pulmonary
– Thrombosis, coronary

Figure 3.5: ACLS Guidelines for Cardiac Arrest Resuscitation

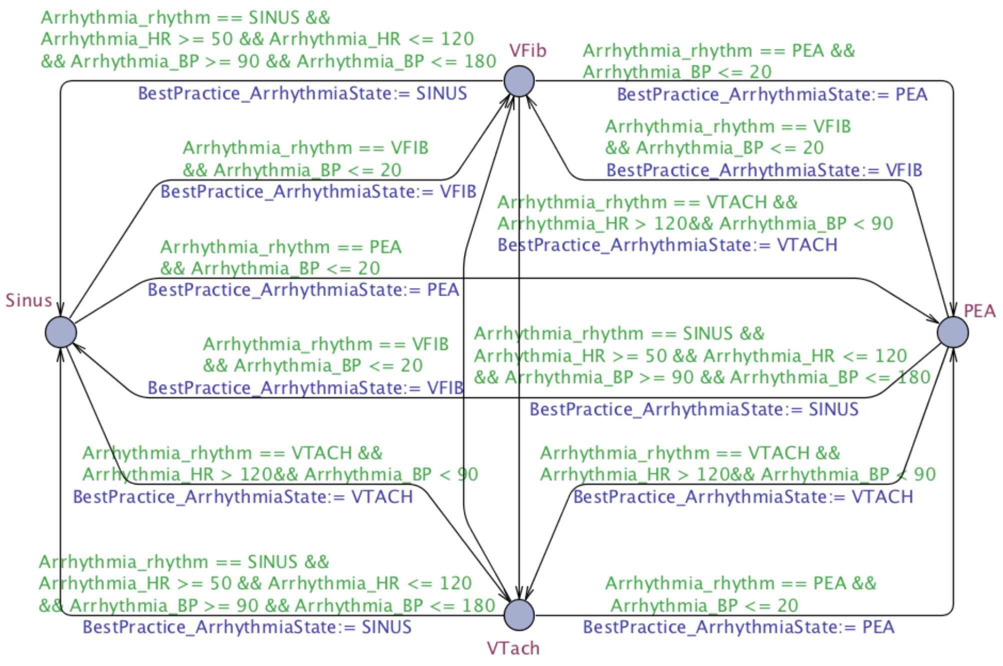
to derive the execution of our cardiac arrest guidance system. However, the clinical correctness of the models must be validated and formally verified before they can be integrated into the guidance system.

3.2 Formal Verification

We use the UPPAAL model checker tool to verify a set of formalized safety properties. We followed the approach explained in Section 2.5, and manually translated all the models to UPPAAL. Figure 3.6(b) shows the *arrhythmia* UPPAAL model, as an example. The *arrhythmia* Yakindu model is also


Arrhythmia
[Rhythm == "Sinus" && HR >= 50 && HR <= 120 && BP >= 90 && BP <= 180]
Ventricular Fibrillation
entry / Communication. sendInfo("VFib is Suspected") entry/BestPractice. ArrhythmiaState = "VFib"
[Rhythm == "PEA" && BP <= 20]
[Rhythm == "VT" && HR >= 120 && BP <= 90]
[Rhythm == "VFib" && BP <= 20]
[Rhythm == "VFib" && BP <= 20]
[Rhythm == "PEA" && BP <= 20]
Sinus
entry/BestPractice. ArrhythmiaState = "Sinus" entry/Communication. sendInfo("Sinus is Suspected")
Pulseless Electrical Activity
entry / Communication. sendInfo("PEA is Suspected") entry/BestPractice. ArrhythmiaState = "PEA"
[Rhythm == "Sinus" && HR >= 50 && HR <= 120 && BP >= 90 && BP <= 180]
[Rhythm == "VT" && HR >= 120 && BP <= 90]
[Rhythm == "VT" && HR >= 120 && BP <= 90]
[Rhythm == "VFib" && BP <= 20]
Ventricular Tachycardia
entry / Communication. sendInfo("VTach is Suspected") entry/BestPractice. ArrhythmiaState = "VTach"
[Rhythm == "Sinus" && HR >= 50 && HR <= 120 && BP >= 90 && BP <= 180]
[Rhythm == "PEA" && BP <= 20]

(a) Yakindu Model


Arrhythmia_rhythm == SINUS && Arrhythmia_HR >= 50 && Arrhythmia_HR <= 120 && Arrhythmia_BP >= 90 && Arrhythmia_BP <= 180
BestPractice_ArrhythmiaState:= SINUS
VFib
Arrhythmia_rhythm == PEA && Arrhythmia_BP <= 20
BestPractice_ArrhythmiaState:= PEA
Arrhythmia_rhythm == VFIB && Arrhythmia_BP <= 20
BestPractice_ArrhythmiaState:= VFIB
Arrhythmia_rhythm == VFIB && Arrhythmia_BP <= 20
BestPractice_ArrhythmiaState:= VFIB
Arrhythmia_rhythm == VTACH && Arrhythmia_HR > 120&& Arrhythmia_BP < 90
BestPractice_ArrhythmiaState:= VTACH
Arrhythmia_rhythm == PEA && Arrhythmia_BP <= 20
BestPractice_ArrhythmiaState:= PEA
Sinus
PEA
Arrhythmia_rhythm == VFIB && Arrhythmia_BP <= 20
BestPractice_ArrhythmiaState:= VFIB
Arrhythmia_rhythm == SINUS && Arrhythmia_HR >= 50 && Arrhythmia_HR <= 120 && Arrhythmia_BP >= 90 && Arrhythmia_BP <= 180
BestPractice_ArrhythmiaState:= SINUS
Arrhythmia_rhythm == VTACH && Arrhythmia_HR > 120&& Arrhythmia_BP < 90
BestPractice_ArrhythmiaState:= VTACH
Arrhythmia_rhythm == VTACH && Arrhythmia_HR > 120&& Arrhythmia_BP < 90
BestPractice_ArrhythmiaState:= VTACH
Arrhythmia_rhythm == SINUS && Arrhythmia_HR >= 50 && Arrhythmia_HR <= 120 && Arrhythmia_BP >= 90 && Arrhythmia_BP <= 180
BestPractice_ArrhythmiaState:= SINUS
Arrhythmia_rhythm == PEA && Arrhythmia_BP <= 20
BestPractice_ArrhythmiaState:= PEA
VTach

(b) UPPAAL Model

Figure 3.6: Arrhythmia Yakindu and UPPAAL Models

Property	Formula
P1	$A[]\ Not\ Deadlock$
P2	$E <> ArrhythmiaBP.VentricularFibrillation$
P3	$A <> ((Rhythm == VFIB\ \&\&\ BP \leq 20) \implies ArrhythmiaBP.VentricularFibrillation)$
P4	$A <> (ArrhythmiaBP.VentricularFibrillation \implies (Rhythm == VFIB\ \&\&\ BP \leq 20))$
P5	$E <> BGI\_BestPractice.MetabolicAcidosis$
P6	$A <> BGI\_BestPractice.MetabolicAcidosis \implies (pH < PH\_LOW\_THRESHOLD)$
P7	$A[]\ BGI\_Physician.MetabolicAcidosis \implies Physician\_BGI\_State == METABOLIC\_ACIDOSIS$
P8	$A <> (BGI\_BestPractice.MetabolicAcidosis\ \&\&\ BGI\_Physician.NormalBloodGasLevels) \implies (MetabolicAcidosis\_DeviationCounter > 0)$

Table 3.1: List of Verified Cardiac Arrest Safety Properties

shown in Figure 3.6(a) to better illustrate the correspondence between the tow models. A subset of validated clinical requirements and several system requirements are formalized as (timed) computation tree logic, CTL, formulas. A subset of verified properties are shown in Table 3.1. The absence of deadlock in the model is formalized as $P1$ and $P2$ is an example of reachability properties. The safety properties $P3$ and $P4$ are examples of formalized state and transition definition correctness properties for arrhythmia model. The property $P7$ ensures the correctness of physician model and $P8$ is a safety property of the divergence-convergence protocol.

3.3 Cardiac Arrest Guidance System

To further demonstrate how the models can be used to design clinical systems, we have integrated our models in a cardiac arrest guidance system following the approach presented in Section 2.6 and illustrated in Figure 2.8. The integration is done using formally verified Yakindu models. The Java code can be generated automatically from the Yakindu model and integrated with the existing system interface using some hand-written glue code. However, the tool also allows direct interaction between the Yakindu model and existing Java code. Therefore, for visualization purposes we have integrated our cardiac arrest model with an existing guidance systems, previously developed by our research group.

All the physiological measurements and relevant lab values can be entered through the guidance system user interface. We have defined a set of events in Yakindu to emulate the physician's response to the system's suggestions; the state machines respond to the activation of these signals (which occurs

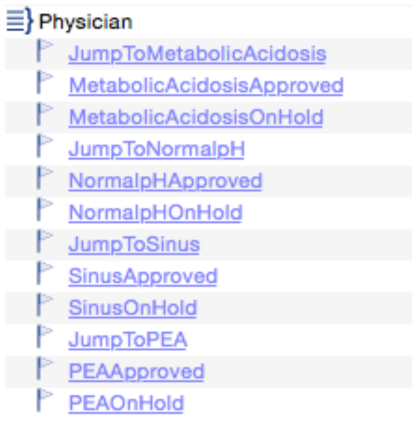

Physician
JumpToMetabolicAcidosis
MetabolicAcidosisApproved
MetabolicAcidosisOnHold
JumpToNormalpH
NormalpHApproved
NormalpHOnHold
JumpToSinus
SinusApproved
SinusOnHold
JumpToPEA
PEAApproved
PEAOnHold

Figure 3.7: Signals Simulating Interaction with Attending Physician

when one clicks on the corresponding signal). A screen shot showing a subset of such signals is shown in Figure 3.7. The threshold values may also be changed in the Yakindu model at both configuration time and during model simulation. If the new patient state is not confirmed or a different organ state is entered by the physician, the convergence-divergence protocol, as explained in Section 2.4.1, is executed. However, if the new patient state is confirmed, the new guidelines are sent to the system to be displayed for medical staff.

Figure 3.8(a) shows a snapshot of our simulation run including the organ models (Statechart implementation) and the graphical display. The snapshot shows a point in the simulation, in which the patient's blood pH value indicates metabolic acidosis. However, the physician does not approve the pathophysiological state suggested by the system. Therefore, Blood Gas Imbalance state machine and its corresponding physician state machine have different runtime states. As shown in Figure 3.8(b), the deviation from best practice guided acidosis diagnosis is logged by the system. Note that when blood pH goes above the threshold, *BGI* state machine automatically transitions to *Normal Blood Gas Levels* state and converges with the physician diagnosis. This case is not shown. When physician approves the pathophysiological states suggested by the system, the subset of best practice guidelines applicable to the current patient state are placed on the guidance system display. Figure 3.9 shows a simulation snapshot, in which PEA rhythm, metabolic acidosis and renal insufficiency states have been approved by physician. As another example, we simulated the case in which there is an inconsistency between correlated measurements. In the previous scenario, where the EKG rhythm indicates ventricular tachycardia, the heart rate and blood pressure values are consistent. On the other hand, Figure 3.10 shows the alert message generated by the system when blood pressure is, according to best practice monitoring guidelines, inconsistent with the PEA (pulseless electrical activity) rhythm. The inconsistent measurement is shown on a red background on the graphic display. For more information on our cardiac arrest developed model and guidance system refer to [67]. The readers can also run their own simulation for the encoded scenarios by accessing the required files [67]. Moreover, a video showing our system simulation for additional scenarios can be accessed here [68].

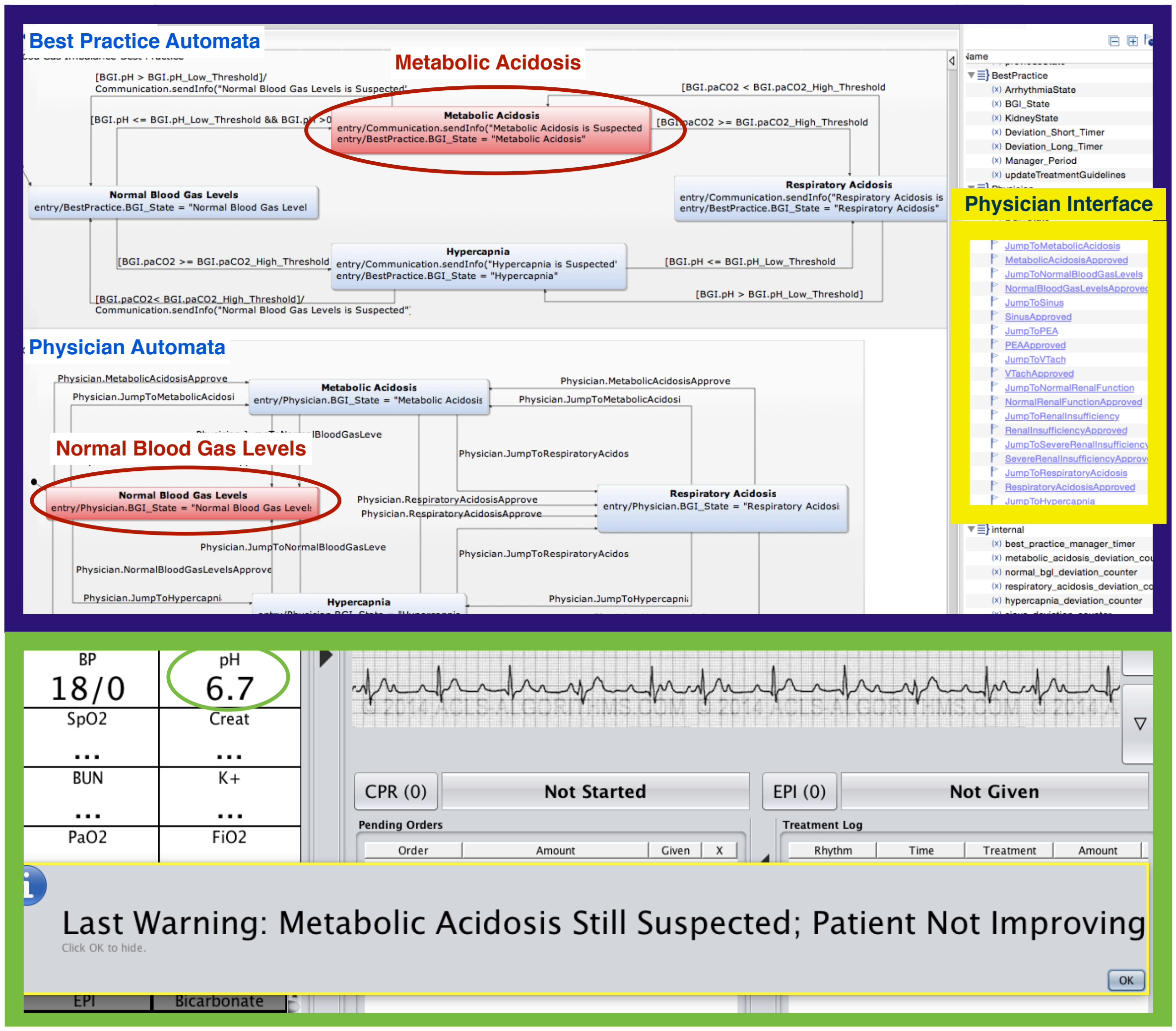
3.4 Clinical Validation

Physicians play the main role in validating the clinical correctness of our models. We also depend on them to assess the usefulness of the *guidance system* in terms of improving *patient state assessment* and *adherence to the best practice guidelines*.

3.4.1 Correctness

We demonstrated our system simulation to a team of ICU medical staff at Carle hospital [69] to ensure clinical correctness of the encoded guidelines such as the ones presented above. The following are a subset of clinical requirements for the cardiac arrest, which are instantiations of the generic requirements presented in Section 2.7.

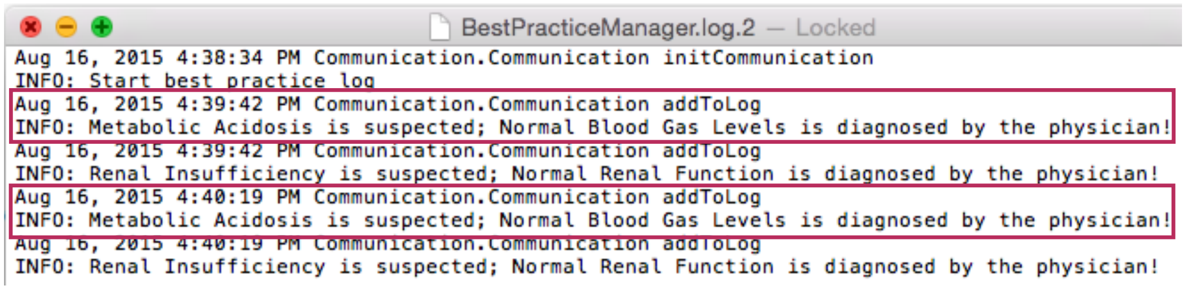
- CR_1: All the pathophysiological processes relevant to cardiac arrest are modeled.
- CR_2: All the relevant abnormal rhythms are modeled in the Arrhythmia machine.
- CR_3: EKG rhythm, HR and BP are the correct and necessary physiological measurements to define different arrhythmia states.
- CR_4: The definition of arrhythmia, BGI and renal insufficiency states are correct.
- CR_5: Any changes in blood pH and/or $PaCO2$ will result in transition to the correct BGI state.


Best Practice Automata
Metabolic Acidosis
[BGI.pH > BGI.pH_Low_Threshold]/ Communication.sendInfo("Normal Blood Gas Levels is Suspected"
[BGI.pH <= BGI.pH_Low_Threshold && BGI.pH >0
Metabolic Acidosis
entry/Communication.sendInfo("Metabolic Acidosis is Suspected
entry/BestPractice.BGI_State = "Metabolic Acidosis"
[BGI.paCO2 < BGI.paCO2_High_Threshold
[BGI.paCO2 >= BGI.paCO2_High_Threshold
Normal Blood Gas Levels
entry/BestPractice.BGI_State = "Normal Blood Gas Level
Respiratory Acidosis
entry/Communication.sendInfo("Respiratory Acidosis is
entry/BestPractice.BGI_State = "Respiratory Acidosis"
[BGI.paCO2 >= BGI.paCO2_High_Threshold
Hypercapnia
entry/Communication.sendInfo("Hypercapnia is Suspected'
entry/BestPractice.BGI_State = "Hypercapnia"
[BGI.pH <= BGI.pH_Low_Threshold
[BGI.paCO2< BGI.paCO2_High_Threshold]/ Communication.sendInfo("Normal Blood Gas Levels is Suspected"
[BGI.pH > BGI.pH_Low_Threshold]
Physician Automata
Physician.MetabolicAcidosisApprove
Physician.JumpToMetabolicAcidosi
Metabolic Acidosis
entry/Physician.BGI_State = "Metabolic Acidosis
Physician.MetabolicAcidosisApprove
Physician.JumpToMetabolicAcidosi
Normal Blood Gas Levels
Physician.JumpToRespiratoryAcidos
Normal Blood Gas Levels
entry/Physician.BGI_State = "Normal Blood Gas Levels
Physician.RespiratoryAcidosisApprove
Physician.RespiratoryAcidosisApprove
Respiratory Acidosis
entry/Physician.BGI_State = "Respiratory Acidosi
Physician.JumpToNormalBloodGasLeve
Physician.NormalBloodGasLevelsApprove
Physician.JumpToRespiratoryAcidos
Physician.JumpToHypercapni
Hypercapnia
Physician.JumpToHypercapnia
Name
BestPractice
ArrhythmiaState
BGI_State
KidneyState
Deviation_Short_Timer
Deviation_Long_Timer
Manager_Period
updateTreatmentGuidelines
Physician Interface
JumpToMetabolicAcidosis
MetabolicAcidosisApproved
JumpToNormalBloodGasLevels
NormalBloodGasLevelsApproved
JumpToSinus
SinusApproved
JumpToPEA
PEAApproved
JumpToVTach
VTachApproved
JumpToNormalRenalFunction
NormalRenalFunctionApproved
JumpToRenalInsufficiency
RenalInsufficiencyApproved
JumpToSevereRenalInsufficiency
SevereRenalInsufficiencyApprov
JumpToRespiratoryAcidosis
RespiratoryAcidosisApproved
JumpToHypercapnia
internal
best_practice_manager_timer
metabolic_acidosis_deviation_cou
normal_bgl_deviation_counter
respiratory_acidosis_deviation_co
hypercapnia_deviation_counter
BP
18/0
pH
6.7
SpO2
Creat
BUN
K+
PaO2
FiO2
CPR (0)
Not Started
EPI (0)
Not Given
Pending Orders
Order
Amount
Given
X
Treatment Log
Rhythm
Time
Treatment
Amount
OK
EPI
Bicarbonate

Last Warning: Metabolic Acidosis Still Suspected; Patient Not Improving

Click OK to hide.

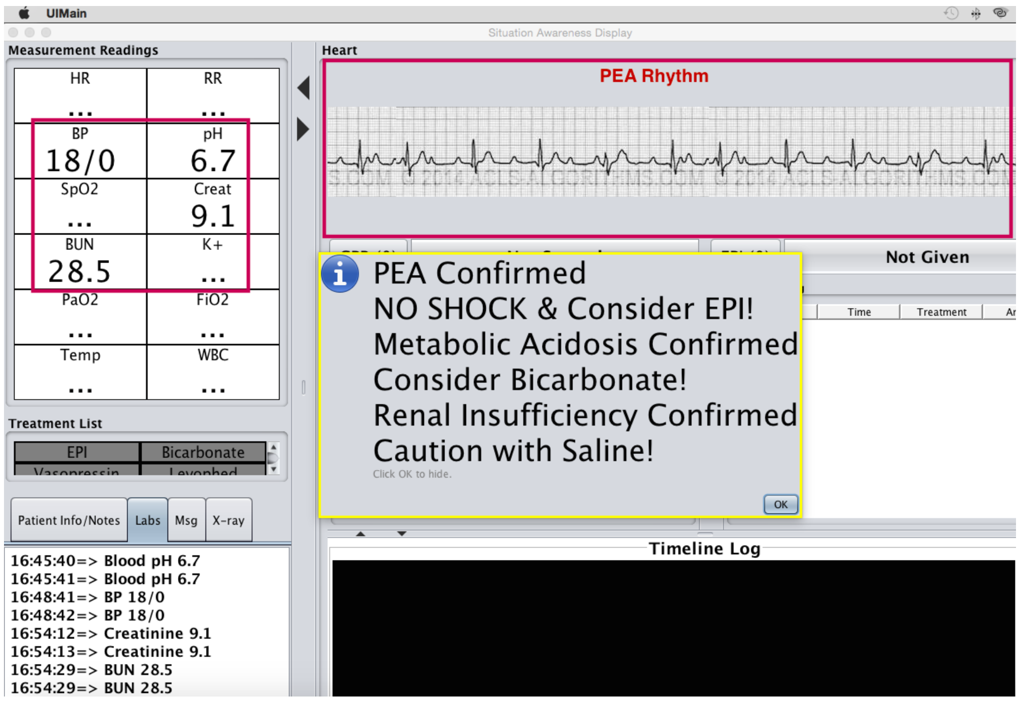
(a) Simulation Snapshot



```
Aug 16, 2015 4:38:34 PM Communication.Communication initCommunication
INFO: Start best practice log
Aug 16, 2015 4:39:42 PM Communication.Communication addToLog
INFO: Metabolic Acidosis is suspected; Normal Blood Gas Levels is diagnosed by the physician!
Aug 16, 2015 4:39:42 PM Communication.Communication addToLog
INFO: Renal Insufficiency is suspected; Normal Renal Function is diagnosed by the physician!
Aug 16, 2015 4:40:19 PM Communication.Communication addToLog
INFO: Metabolic Acidosis is suspected; Normal Blood Gas Levels is diagnosed by the physician!
Aug 16, 2015 4:40:19 PM Communication.Communication addToLog
INFO: Renal Insufficiency is suspected; Normal Renal Function is diagnosed by the physician!
```

(b) Log Snapshot

Figure 3.8: Inconsistent Physician and Best Practice Blood Gas Imbalance Automata



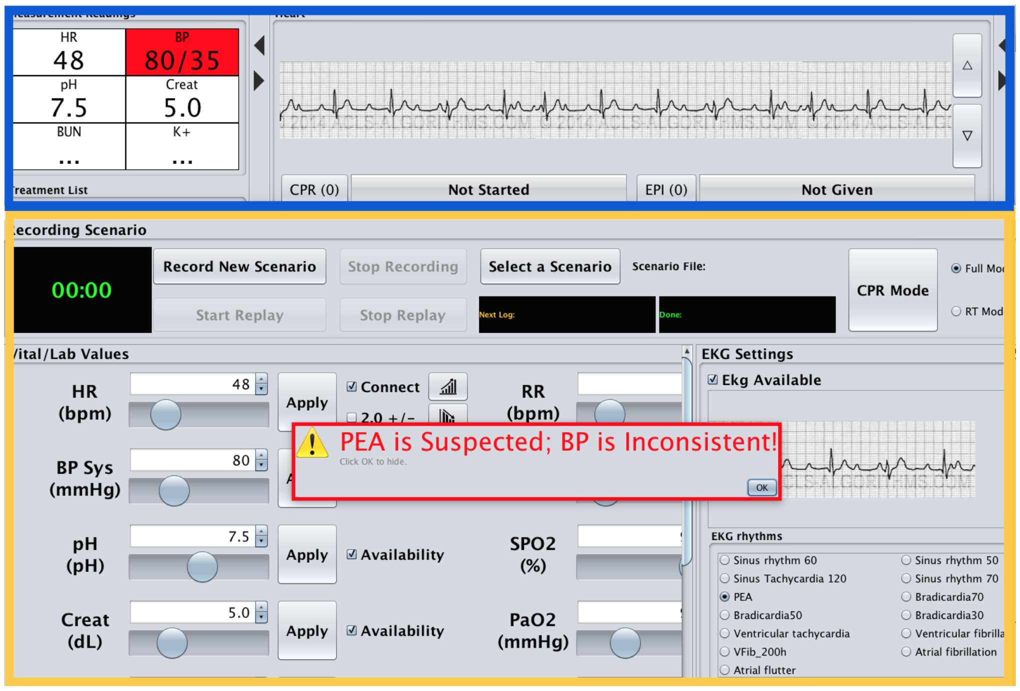
HR ...	RR ...
BP 18/0	pH 6.7
SpO2 ...	Creat 9.1
BUN 28.5	K+ ...
PaO2 ...	FiO2 ...
Temp ...	WBC ...

EPI	Bicarbonate
Vasopressin	Levophed

PEA Confirmed
NO SHOCK & Consider EPI!
Metabolic Acidosis Confirmed
Consider Bicarbonate!
Renal Insufficiency Confirmed
Caution with Saline!

```
16:45:40=> Blood pH 6.7
16:45:41=> Blood pH 6.7
16:48:41=> BP 18/0
16:48:42=> BP 18/0
16:54:12=> Creatinine 9.1
16:54:13=> Creatinine 9.1
16:54:29=> BUN 28.5
16:54:29=> BUN 28.5
```

Figure 3.9: Best Practice Guidelines Generated by our System



HR 48	BP 80/35
pH 7.5	Creat 5.0
BUN ...	K+ ...

CPR (0) Not Started
EPI (0) Not Given

00:00

[✓ Full Mode] RT Mode

HR (bpm) 48 — ☑ Connect ☐ 2.0 +/-
BP Sys (mmHg) 80
pH (pH) 7.5 — ☑ Availability
Creat (dL) 5.0 — ☑ Availability

☑ Ekg Available

PEA is Suspected; BP is Inconsistent!

Sinus rhythm 60	Sinus rhythm 50
Sinus Tachycardia 120	Sinus rhythm 70
[✓ PEA]	Bradicardia70
Bradicardia50	Bradicardia30
Ventricular tachycardia	Ventricular fibrilla
VFib_200h	Atrial fibrillation
Atrial flutter	

Figure 3.10: EKG Rhythm and Blood Pressure Inconsistency Alert

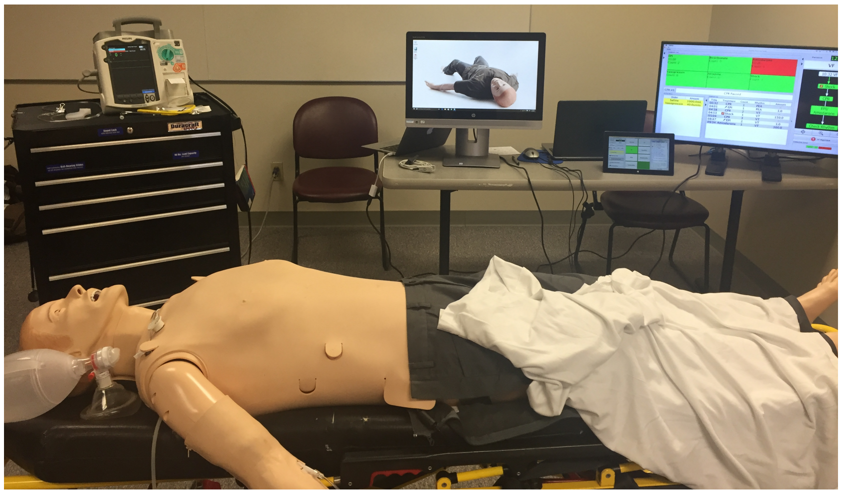


Figure 3.11: Medical SimMan 3G Manikin in Carle ACLS Training Class

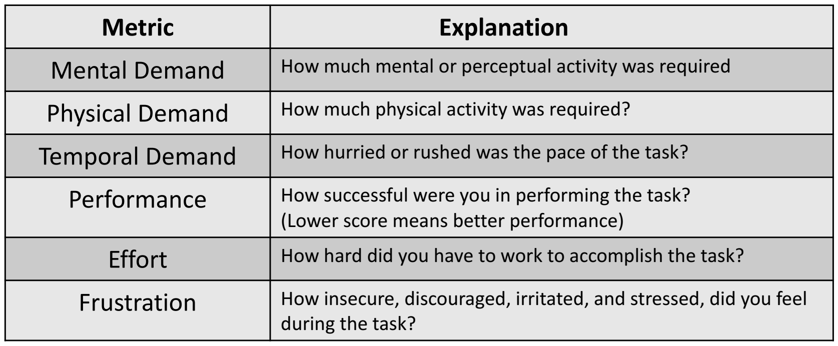
3.4.2 Usefulness

One of challenges encountered in developing any guideline model or designing clinical guidance system is to determine relevant medical knowledge to be encoded in the model. Furthermore, identifying what guideline items and at which time to be presented to clinical staff is critical. These are deciding factors for usability of the system in actual clinical environment or for the purpose of training nurses or residents. We conducted a preliminary evaluation of our cardiac arrest guidance system in a ACLS training class with 18 medical personnel consisting of physicians and nurses from Carle Foundation Hospital.

Note that there are two versions of cardiac arrest guidance system. The version presented in this thesis is designed based on organ-centric pathophysiology-based model can be simulated and evaluated before code generation. The first version was developed using traditional procedural style approaches. There is also a 2-part study protocol in place in collaboration with Carle hospital ICU department.These studies will evaluate our cardiac arrest guidance system in terms of usability, recording accuracy and improvement in adherence to ACLS workflow; and compare them with the traditional approach of memorizing workflows and using paper and pencil for recording.

Using our model-driven cardiac arrest guidance4 system, we conducted two sets of simulations on the medical SimMan 3G Manikin [3], which is used for training purpose (Figure 3.11). One set of simula-

[3]http://www.laerdal.com/us/SimMan3G



Metric	Explanation
Mental Demand	How much mental or perceptual activity was required
Physical Demand	How much physical activity was required?
Temporal Demand	How hurried or rushed was the pace of the task?
Performance	How successful were you in performing the task? (Lower score means better performance)
Effort	How hard did you have to work to accomplish the task?
Frustration	How insecure, discouraged, irritated, and stressed, did you feel during the task?

Figure 3.12: NASA Task Load Index Metrics

tions was done with our guidance system and one without the system. Each set includes three different cardiac arrest scenarios. In each simulation, a medical team consists of four to eight members follow ACLS algorithm to resuscitate a cardiac arrest patient , whose state dynamically changes according to the scenario. The resuscitation process includes managing patient's airway and breathing, diagnosing the rhythm, performing CPR, shock therapy and administration of constrictive medication.

After the medical team finished the two scenarios, they evaluated their workload using the six metrics in NASA Task Load Index (NASA-TLX) questionnaire. The metrics and their descriptions is shown in Figure 3.12. NASA-TLX [70] is a commonly used subjective assessment tool that rates perceived workload in order to evaluate users performance. It has been used in various domains, including aviation and healthcare. The physicians and nurses gave a score from 1 to 20 for each metric, and the lower score means the lower demand, better performance, and less effort and frustration.

Figure 3.13 shows the average score for each of the six metrics from the 18 physicians and nurses. Our system greatly reduces the mental demand and temporal demand, because the relevant clinical information which is critical to assessment of patient is provided by the system. This makes easier for medical staff to track patient's state and response to treatments in realtime. Moreover, the medical staff believed the guidance system help them with their performance. In addition, the effort and frustration levels are also low, which suggests that the displayed information is critical and easy for medical staff to comprehend.

We also ask them to rate the usefulness of different guideline items provided by the system, using a similar 20-point scoring system. A higher score means a more useful guideline item. The following are the list of specific guideline items included in this evaluation:

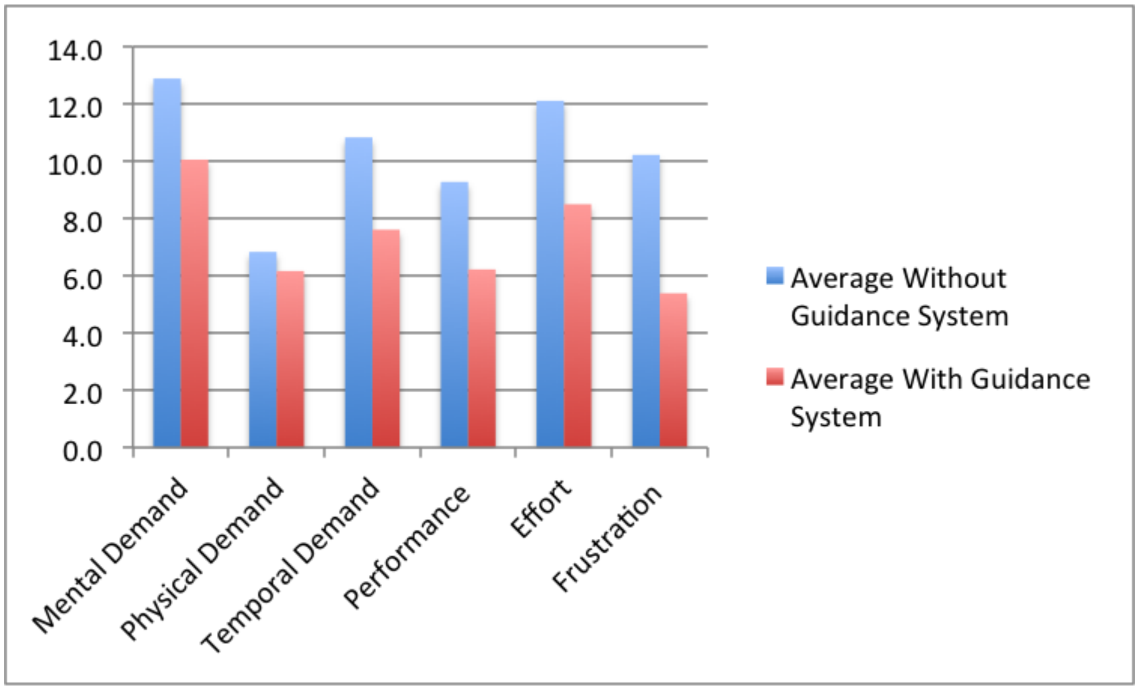


Figure 3.13: NASA TLX Evaluation Results from 18 Clinicians

1. Alerts on inconsistencies between the diagnosed rhythm and the physiological measurements (Ex: VT rhythm and heart rate = 80 bpm)

2. Alerts on out-of-range drug dosage (Ex: Order 5.0 mg of EPI)

3. Alerts on total drug injection count (Ex: Inject EPI more than 3 times)

4. Alerts on checking ETCO2 when getting close to the end of 2 minutes of CPR

5. Alerts on treatment contraindications (Ex1:Alert on low pH, when EPI is ordered, Ex2: Alert on Non-shockable rhythm when Shock is ordered)

6. Alerts on treatment guidelines when patient condition changes (Ex1: Consider injecting Bicarbonate, when pH is below 7.2, Ex2: Shock when the rhythm is VT or VF)

Table 3.2 shows the average score for each guideline item according to the rating from 11 clinicians. All average scores are above 17, which indicates that the clinical users in the ACLS training class found these guideline items very useful. We have conducted this evaluation to specifically address the second part of design **Challenges 6**, that is assessing the effectiveness and safety of any proposed alarm

Guideline Item	Avergae Usefulness Score
Measurement Inconsistency Alert	17.2
Out-of-Range Drug Dosage Alert	17.9
Total Drug Injection Count Alert	18.1
ETCO2 Check Alert	18.3
Treatment Contraindications Alert	17.7
Treatment Guidelines Alert	17.3

Table 3.2: Guideline Usefulness Evaluation Results from 11 Clinicians

management. The first part of this challenge describes the necessity of an alarm management approach to reduce slips and lapse caused by inadequate alarm configuration and practices.

Another goal was evaluating the usefulness of our communication model. According to [5], two styles of communication are often followed in clinical guidance systems: *consulting* model vs. *critiquing* model. In a consulting model the system actively provides monitoring and treatment guidelines. However, in a critiquing model physician already has a plan in place, possibly with a set of goals expressed in terms of patient state. The system only generates alerts or provide other guideline items when the physician is deviating from the plan or moving further away from the goal.

Our model-driven guidance system follows a combination of these two communication models. We believe, in some instances clinical staff prefer the second model; we have verified this to some degree in our evaluation. The challenge is how to determine when to use each communication model. We found out that one determining factor is the level of experience of the clinical user. We use this information and applied it to our sepsis guidance system. Three guidance levels are defined in our system: *high* for inexperience users, *medium* for users with some degree of experience such as residents, and *low* for the most experienced users. The guideline items are flagged with a level as well. The clinical user can choose a guidance level and only the guideline items corresponding to chosen level will be displayed. The other factors can be disease and the procedure. For example, in our evaluation of cardiac arrest guidance system, we found out that for some procedures such as CPR, drug administration, or shock therapy the clinical users prefer a critiquing communication model. They do not want to be bothered with message such as ”The rhythm is VTach; Consider Shock”; however, if for any reason they are about to shock a patient with PEA rhythm, they find an alert to help them avoid mistakingly shocking the patient useful. On the other hand, in the case of sepsis screening and management some clinical staff ay need some handling during the care process and therefore, prefer a consulting communication model. However, we should note that these findings are only based on our preliminary evaluation and further studying of different case studies required to draw a more thorough and clear specific conclusions.

The overall lesson learned is that only adequate but not excessive guideline data for making informed decision must be presented to clinical users. Generating excessive number of warnings for minor problems must be avoided. Providing more focused patient-specific information and flagging abnormal things which otherwise would have been overlooked are steps into right direction. Providing excessive information prevents the clinical staff form processing and synthesizing information intelligently and rapidly. This is specifically important for intensive care units, since they can be considered the classic example of information overload.

We also demonstrated our system simulation to a team of ICU medical staff at Carle hospital [69] to ensure clinical correctness of the encoded guidelines such as the ones presented above. They also found several aspects of our system specifically helpful, including the following:

- Organizing the available measurements and relevant clinical data according to an organ-centric structure. For example, our kidney state machine uses BUN, creatinine, and potassium to define different states of renal insufficiency. In addition to displaying individual measurements, our system also informs the physician of the severity of renal insufficiency, determined according to best practice monitoring guidelines [66]. Codifying a layer of intelligence on top of raw measurements allows our system to provide more relevant contextual information about patient's condition (in terms of current state of different organ systems). Our physician collaborators find the structured contextual information, provided by the system, very useful.

- Informing the medical staff of any additional complications the cardiac arrest patient may develop. For example, generating alerts when acidosis or renal insufficiency is suspected, while epinephrine is being administered, can be very useful. Note that the *best practice assist system* determines such insufficiency states according to the best practice guidelines encoded in our models. Medical staff can use this information to adjust the treatment plan and adhere to the best practice guidelines.

- Raising an alert when there is an inconsistency between correlated measurements. Previously, we presented an example, in which the system generates an alert message informing the medical staff of an inconsistency between an EKG waveform indicating PEA and a blood pressure value of greater than $20mmHg$.

At the end of this section we include a direct quote from our ICU physician collaborator:

> Healthcare providers are burdened at the patient's bedside by a wealth of information, often from separate and disparate sources. The physician in particular must review, digest, assimilate and apply increasing amounts of information in an ever-shortening time frame, and identify which specific best practice guidelines should be used. Cardiac arrest is the obvious example where multiple monitors, sensors, rhythms, lab values characterize the patient care situation. Order must be brought to this enlarging problem as medical science progresses. The organ based automata approach to software design within an assistant hierarchy is a logical solution. Such a systematic structure mirrors how health care providers learn and understand human disease. The development of these architectures will provide medicine, especially acute care at the bedside, the computational interface that is needed.

3.5 Research Publications

The proposed system integration approach, and the result of our preliminary clinical evaluation of cardiac arrest guidance system in the ACLS training class has led to the following submission:

- **M. Rahmaniheris**, Y. Jiang, P. Wu, and L. Sha, *Design and Evaluation of Model-Driven Guidance Systems for Acute Care*" to be submitted to IEEE Real-Time Systems Symposium (RTSS) 2017

Chapter 4

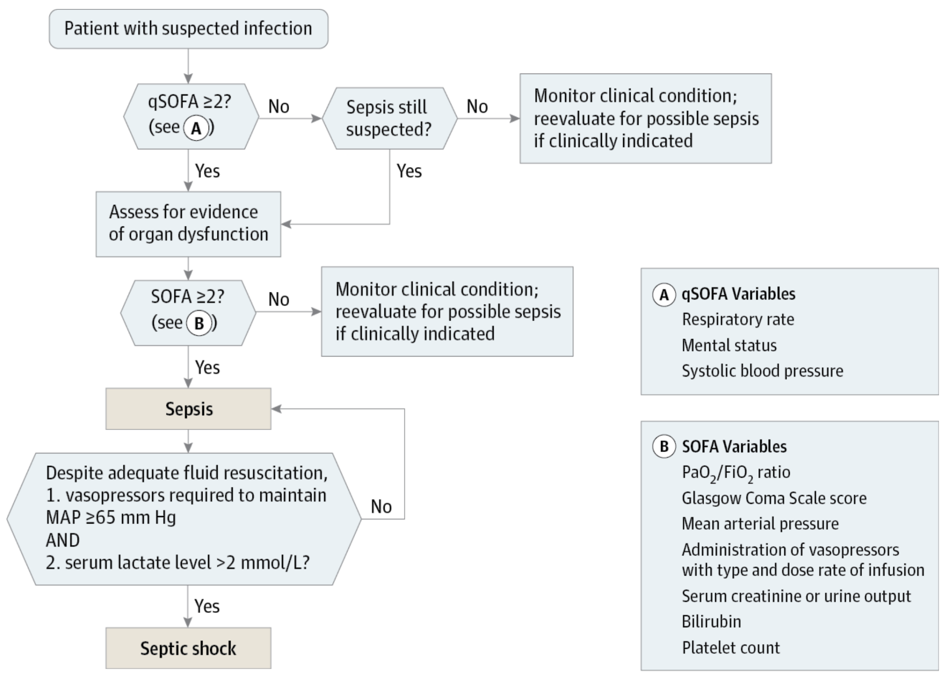
Sepsis Case Study

Sepsis is defined as life-threatening organ dysfunction caused by a dysregulated host response to infection [71]. Early identification of sepsis helps increase the chance of timely interventions to treat the cause of infection and manage sepsis. This improves the effectiveness of evidenced based therapies and can greatly improve the survival rate. However, if it is not detected in early stages, the patient can develop septic shock, which is defined as a subset of sepsis in which particularly profound circulatory abnormalities are associated with a greater risk of mortality than with sepsis alone. Figure 4.1 illustrates the clinical criteria used for sepsis screening. This is according to the most recent definition of sepsis [71]. According to surviving sepsis campaign [72], there are three main steps in the process of diagnosis and treatment:

- **Step 1: Screening and Management of Infection**. This step focuses on management of patients identified as having infection by obtaining blood and other cultures as indicated, and administering tailored antibiotics as appropriate.

- **Step 2: Screening for Organ Dysfunction and Management of Sepsis**. If organ dysfunction identified, sepsis workflow items, such as lactate measurement and fluid resuscitation, must be initiated.

- **Step 3: Identification and Management of Initial Hypotension**. In addition to fluid resuscitation, vasopressors must be administered to reach and maintain a normal mean arterial pressure.


Patient with suspected infection
qSOFA ≥2? (see A)
No
Sepsis still suspected?
No
Monitor clinical condition; reevaluate for possible sepsis if clinically indicated
Yes
Yes
Assess for evidence of organ dysfunction
SOFA ≥2? (see B)
No
Monitor clinical condition; reevaluate for possible sepsis if clinically indicated
Yes
Sepsis
Despite adequate fluid resuscitation, 1. vasopressors required to maintain MAP ≥65 mm Hg AND 2. serum lactate level >2 mmol/L?
No
Yes
Septic shock
A qSOFA Variables
Respiratory rate
Mental status
Systolic blood pressure
B SOFA Variables
PaO2/FiO2 ratio
Glasgow Coma Scale score
Mean arterial pressure
Administration of vasopressors with type and dose rate of infusion
Serum creatinine or urine output
Bilirubin
Platelet count

Singer, Mervyn, et al. "*The third international consensus definitions for sepsis and septic shock (sepsis-3)*." Jama 315.8 (2016): 801-810.

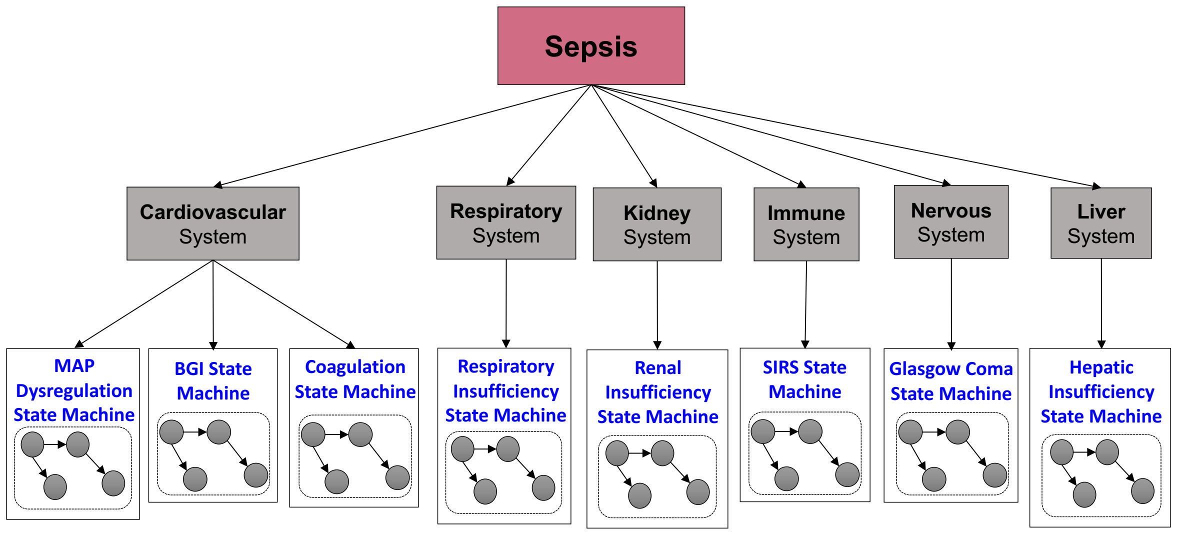
Figure 4.1: Clinical Criteria for Sepsis Screening

In the next section, we present the organ-centric pathophysiology-based sepsis model. We follow the guidelines presented in Section 2.3 to construct the models.

4.1 Model Construction

We start by applying our first modeling guideline, $\mathbf{G_1}$. According to this guideline, a pathophysiological process becomes relevant if the corresponding physiological function is affected by the disease or affect the treatment. Figure 4.1 shows the main organ systems and physiological functions that may be affected by sepsis. For example, mean arterial pressure (MAP) and coagulation insufficiencies are examples of cardiovascular abnormalities. Therefore MAP dysregulation and coagulation insufficiencies are relevant pathophysiological processes that must be tracked when monitoring and treating a cardiac arrest patient. The others are identified in a similar way. Figure 4.2 shows the complete list of sepsis pathophysiological processes.

Next, we follow $\mathbf{G_2}$ and identify the information that is important, but not need to be encoded in the same details as others in our sepsis model. As demonstrated in Figure 4.1, more cardiovascular



Sepsis
Cardiovascular System
Respiratory System
Kidney System
Immune System
Nervous System
Liver System
MAP Dysregulation State Machine
BGI State Machine
Coagulation State Machine
Respiratory Insufficiency State Machine
Renal Insufficiency State Machine
SIRS State Machine
Glasgow Coma State Machine
Hepatic Insufficiency State Machine

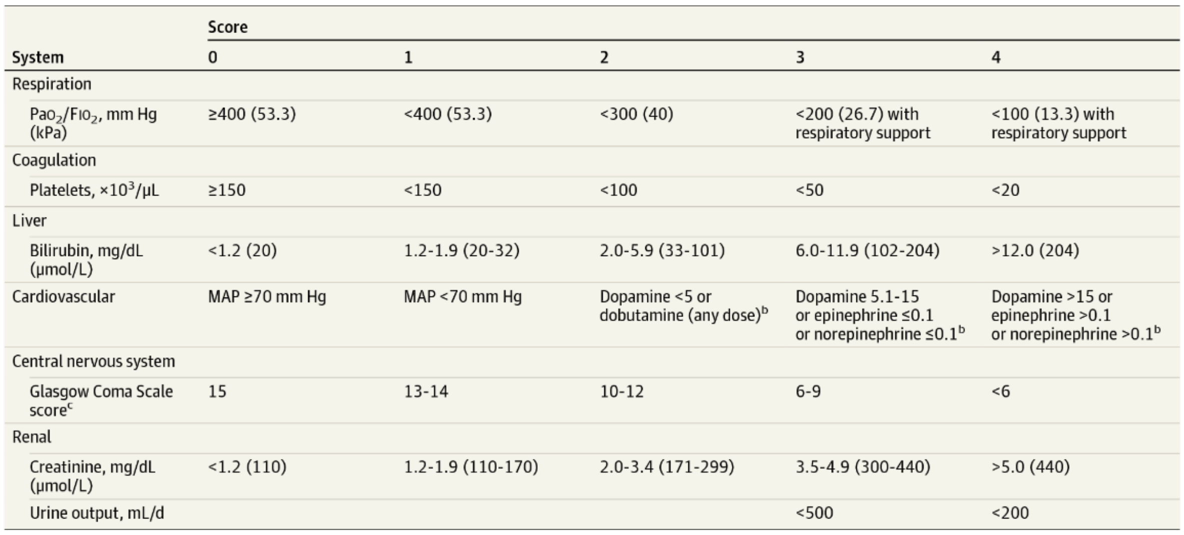
Applied Modeling Guidelines
G1: **Relevant** pathophysiological processes for the disease
G2: **Aggregating** important not needed in the same detail as others
G3: Modeling the simultaneous effects by **concurrent state machines**.

Figure 4.2: Sepsis Pathophysiological State Machines

functions are affected and they all must be tracked for sepsis management. In other words, it is not simply sufficient to represent all these changes by an aggregated degree of insufficiency. On the other hand, clinical information regarding some other organs, such as respiratory and kidney systems, are aggregated as the degree of insufficiency.

At this stage, we must determine the concurrency level of our sepsis model. According to $\mathbf{G_3}$, the affected physiological functions that need to be tracked simultaneously must be modeled by separate finite state machines. If $\mathbf{G_2}$ has already been applied, following this guideline becomes straightforward. Previously, we specified the different (aggregated) pathophysiological states need to be tracked for a septic patient. To keep track of all these states and prevent any critical changes from being overlooked we model each process using a separate state machine. These state machines are show in Figure 4.2, which demonstrate the result of applying the modeling guidelines $\mathbf{G_1}$ - $\mathbf{G_3}$.

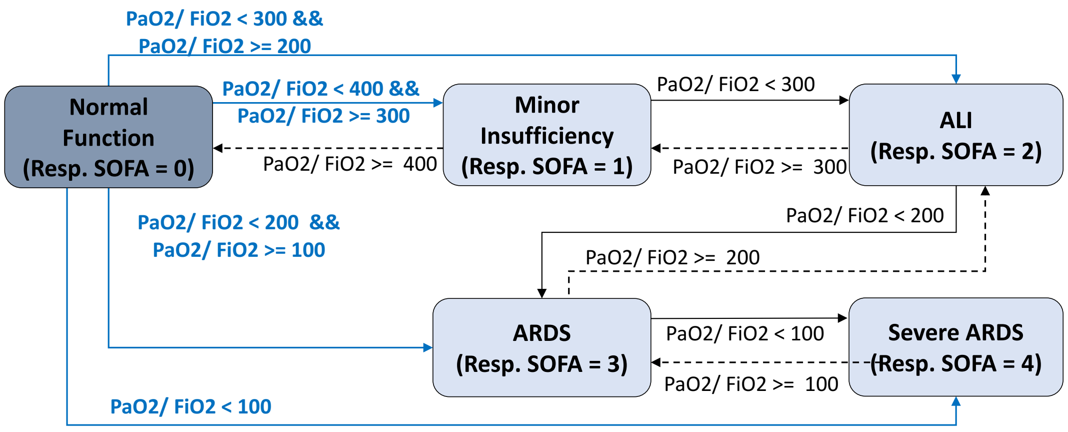
Now, must define the states for each of the identified pathophysiological state machines. As shown in 4.1, sequential organ failure assessment (SOFA) score is used to represent how and to what degree each organ function is affected [60]. The scores are calculated based on the current values of the



	Score				
System	0	1	2	3	4
Respiration					
PaO_2/FiO_2, mm Hg (kPa)	≥400 (53.3)	<400 (53.3)	<300 (40)	<200 (26.7) with respiratory support	<100 (13.3) with respiratory support
Coagulation					
Platelets, $\times 10^3/\mu L$	≥150	<150	<100	<50	<20
Liver					
Bilirubin, mg/dL (μmol/L)	<1.2 (20)	1.2-1.9 (20-32)	2.0-5.9 (33-101)	6.0-11.9 (102-204)	>12.0 (204)
Cardiovascular	MAP ≥70 mm Hg	MAP <70 mm Hg	Dopamine <5 or dobutamine (any dose)[b]	Dopamine 5.1-15 or epinephrine ≤0.1 or norepinephrine ≤0.1[b]	Dopamine >15 or epinephrine >0.1 or norepinephrine >0.1[b]
Central nervous system					
Glasgow Coma Scale score[c]	15	13-14	10-12	6-9	<6
Renal					
Creatinine, mg/dL (μmol/L)	<1.2 (110)	1.2-1.9 (110-170)	2.0-3.4 (171-299)	3.5-4.9 (300-440)	>5.0 (440)
Urine output, mL/d				<500	<200

Vincent, J-L., et al. "The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure." *Intensive care medicine* 22.7 (1996): 707-710.

Figure 4.3: Sequential Organ Failure Assessment

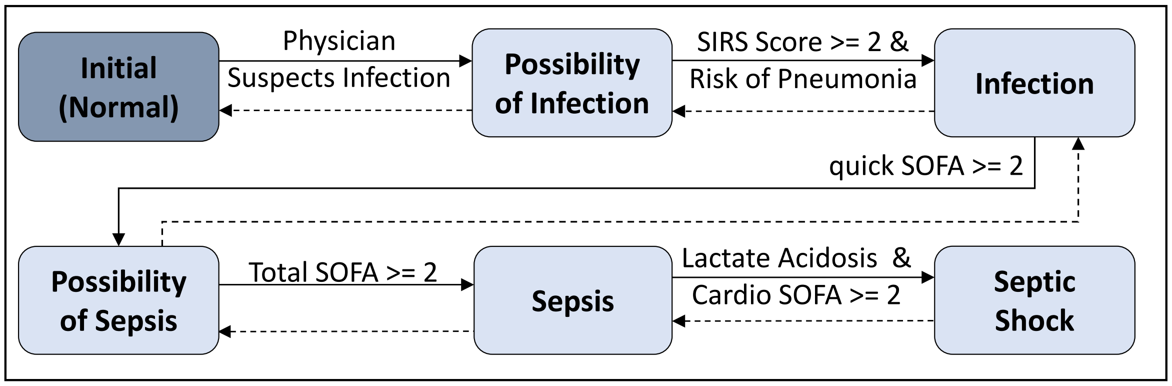


Applied Modeling Guidelines
G4: Defining **actionable** states: each state represents a critical change in patient state and requires a monitoring or treatment action must be performed
G5: Defining **optimal number of transitions:** the transitions indicate the gradual progress of disease (**except the initial state**) and outgoing transitions of each state are mutually exclusive.

Figure 4.4: Respiratory Insufficiency State Machine

corresponding SOFA variable(s).

However, defining a large number of states can make the model very complex and unmanageable. In guideline $\mathbf{G_4}$, we introduced the notions of *actionable states*. Not all the changes in physiological

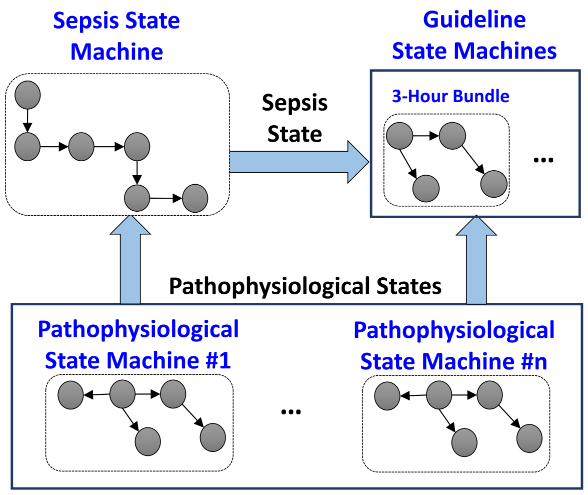


Applied Modeling Guidelines
G4: Defining **actionable** states
G5: Defining **optimal number of transitions**
G6: Codifying the pathophysiological state machines' interactions in a **disease model**

Figure 4.5: Sepsis State Machine

measurements are relevant or useful for a given disease. We define a new state when there is a need for new monitoring and therapeutic plan. The actionable states for sepsis can be directly extracted from the SOFA table shown in Figure 4.3. The minimum SOFA score is 0, which indicate the normal function. The maximum score is 4 indicating severe insufficiency in the corresponding organ function. Therefore, for the pathophysiological state machines that use SOFA score as the their state variables, we define five states, one for each SOFA score. Figure 4.4 shows *respiratory insufficiency* state machine, as an example.

After defining the states, we must identify the correct and optimal number of transitions by considering the underlying medical knowledge. According to guideline $\mathbf{G_5}$, when continuous physiological variables, such as SOFA scores, are used to define different states, the transitions are only defined to represent the progress of insufficiency. We only define an edge between the states that represent gradual deterioration and improvement. Figure 4.4 shows this guidelines is applied to *respiratory insufficiency* state machine. As mentioned before, the initial state is the exception, since patient state is not know at the start.


Sepsis State Machine
Guideline State Machines
3-Hour Bundle
...
Sepsis State
Pathophysiological States
Pathophysiological State Machine #1
...
Pathophysiological State Machine #n

Applied Modeling Guideline
G6: Minimizing the **interaction** between the pathophysiological state machines

Figure 4.6: Sepsis Model Interaction Diagram

Following the last guideline, $\mathbf{G_6}$, we do not define any interaction between different pathophysiological state machines. We codify the potential interactions a sepsis disease model, as shown in Figure 4.5. The sepsis state machine codifies the different stages of sepsis, starting with potential infection. The information from different pathophysiological processes are aggregated in this machine. Figure 4.6 shows the interaction diagram for the sepsis model. We try to minimize any communication or dependency between the pathophysiological machines. This improves the *model modularity*. The complexity of validation and verification may be reduced since most of pathophysiological state machines can be validated and verified independently of the others.

4.1.1 Sepsis Guideline Model

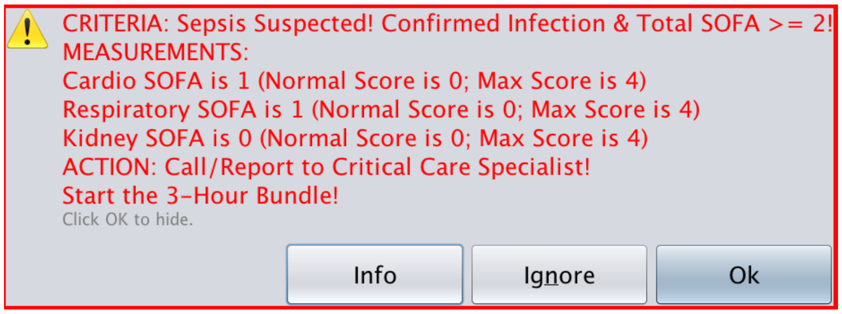
The guideline model must encode the relevant clinical data, that is best practice monitoring and treatment guidelines organized for different pathophysiological and disease states. We use the notion of ***actionable alerts***, which follows the same line of reasoning as *actionable states*, which was describe in Section 2.3. In fact, by encoding the alerts on patient state changes in already defines actionable states we ensure that they are comprehensive and integrated with pathophysiological sates. These alerts may also be issued on patient's response to treatments or best practice treatment option for the current pathophysiological states.

The alerts on development or resolution of organ insufficiency states are encoded in individual pathophysiological states. These alerts only depend on the information from the corresponding pathophysiological process. There are no interaction between different pathophysiological processes and therefore these alerts can be encoded independent of each other. For example, the alert indicating patient has developed moderate renal insufficiency is encoded in the renal state machine.

However, the alerts on progress of the sepsis is encode in the state state machine. These alerts require information from several pathophysiological processes. This aggregated information is encoded in sepsis state machine, which is the best place to encode this category of alerts. The alert indicating a septic patient has deteriorated to septic shock is an example alert from this category. The alert messages, in general, has the following main items:

- The new sepsis/pathophysiological state (and the corresponding process).
- pathophysiological state change type: improvement or deterioration.
- The current value of relevant measurements.
- The normal range for each of the above measurements.
- Recommended treatment and monitoring actions.

Figure 4.7 shows the alert message when sepsis is detected. The information presented in an alert message is added and displayed permanently in a history table on the system's user interface.



CRITERIA: Sepsis Suspected! Confirmed Infection & Total SOFA >= 2!
MEASUREMENTS:
Cardio SOFA is 1 (Normal Score is 0; Max Score is 4)
Respiratory SOFA is 1 (Normal Score is 0; Max Score is 4)
Kidney SOFA is 0 (Normal Score is 0; Max Score is 4)
ACTION: Call/Report to Critical Care Specialist!
Start the 3-Hour Bundle!
Click OK to hide.

Figure 4.7: Respiratory Insufficiency Alert Message

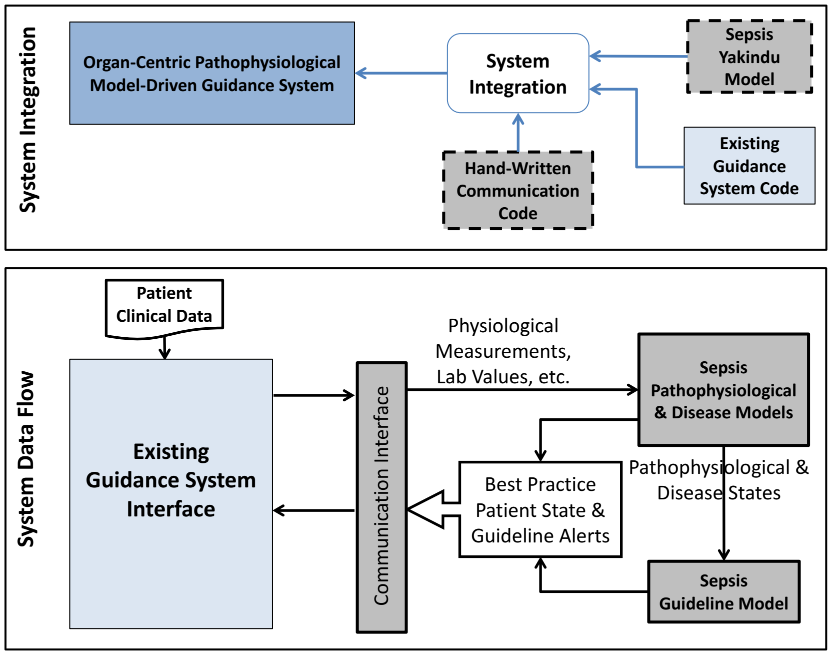
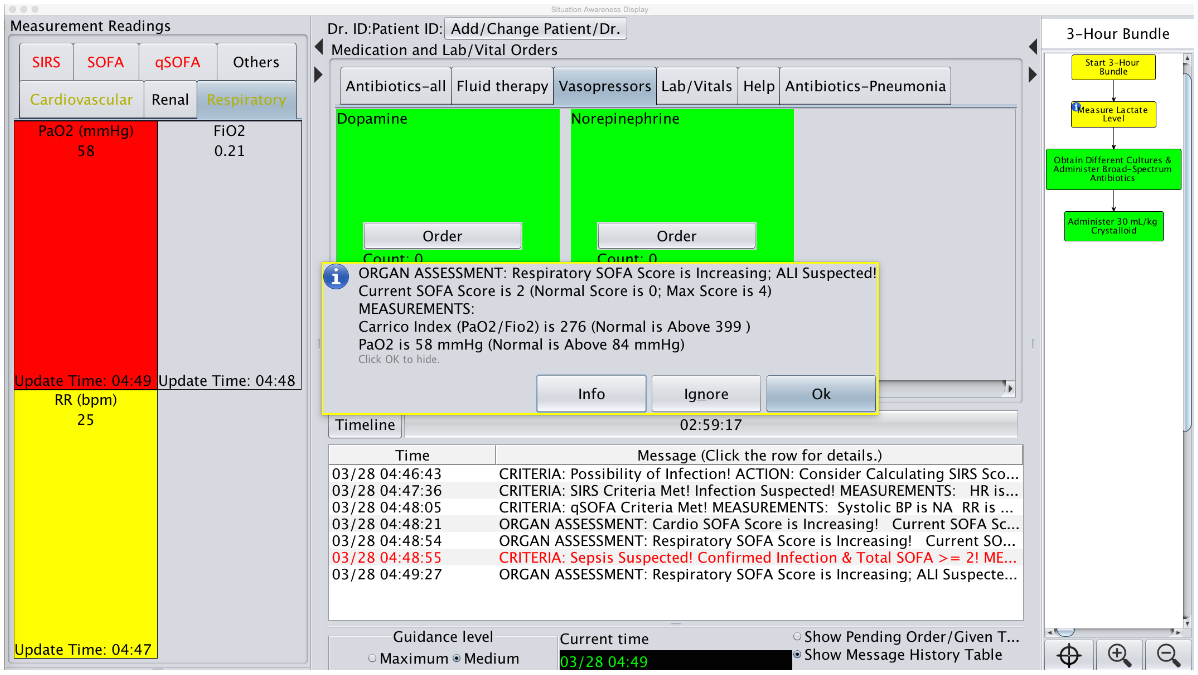

System Integration
Organ-Centric Pathophysiological Model-Driven Guidance System
System Integration
Sepsis Yakindu Model
Existing Guidance System Code
Hand-Written Communication Code
System Data Flow
Patient Clinical Data
Existing Guidance System Interface
Communication Interface
Physiological Measurements, Lab Values, etc.
Sepsis Pathophysiological & Disease Models
Pathophysiological & Disease States
Best Practice Patient State & Guideline Alerts
Sepsis Guideline Model

Figure 4.8: Sepsis Guidance System Integration and Data Flow

4.2 Model Integration

We have developed the initial version of our model-based sepsis guidance system (we only have one version vs. the two versions of cardiac arrest guidance systems). Figure 4.8 shows the system integration process as well the data flow between the sepsis models and the guidance. As described in Section 2.6, we use the modeling tool Yakindu and we have developed a communication interface that allows the sepsis Yakindu models to communicate with the guidance system Java code. The model integration also provides a more user-friendly environment for simulation of different clinical scenarios. This would allow physicians to validate the correctness of developed models more efficiently.

All the physiological measurements and relevant lab values can be entered through the guidance system user interface and then communicated to sepsis models using our hand-written communication code. The pathophysiological and sepsis state machines are updated when new clinical data becomes available, to represent patient current state. A new set of alert messages may be generated as a result of updated patient organ and disease state. Alerts generated by the model are sent to the guidance system to be displayed foe medical staff. Figure 4.9 shows a snapshot of sepsis guidance system simulation displaying an alert message indicating the deterioration of respiratory insufficiency state.



ORGAN ASSESSMENT: Respiratory SOFA Score is Increasing; ALI Suspected!
Current SOFA Score is 2 (Normal Score is 0; Max Score is 4)
MEASUREMENTS:
Carrico Index (PaO2/Fio2) is 276 (Normal is Above 399)
PaO2 is 58 mmHg (Normal is Above 84 mmHg)
Click OK to hide.

Time	Message (Click the row for details.)
03/28 04:46:43	CRITERIA: Possibility of Infection! ACTION: Consider Calculating SIRS Sco...
03/28 04:47:36	CRITERIA: SIRS Criteria Met! Infection Suspected! MEASUREMENTS: HR is...
03/28 04:48:05	CRITERIA: qSOFA Criteria Met! MEASUREMENTS: Systolic BP is NA RR is ...
03/28 04:48:21	ORGAN ASSESSMENT: Cardio SOFA Score is Increasing! Current SOFA Sc...
03/28 04:48:54	ORGAN ASSESSMENT: Respiratory SOFA Score is Increasing! Current SO...
03/28 04:48:55	CRITERIA: Sepsis Suspected! Confirmed Infection & Total SOFA >= 2! ME...
03/28 04:49:27	ORGAN ASSESSMENT: Respiratory SOFA Score is Increasing; ALI Suspecte...

Figure 4.9: Sepsis Guidance System Simulation

4.3 Remaining Challenges

Identifying at-risk patients as early as possible while maintaining a reasonable monitoring cost, is critical for effective sepsis screening and management. Performing frequent monitoring of all patients suspected of sepsis is not feasible and cost-effective. Therefore, it is critical to identify the patients at a higher risk of sepsis and perform more extensive and frequent monitoring only for this class of patients.

As part of our future work, we plan on developing a risk driven sepsis model encoding dynamic Bayesian analysis in different pathophysiological states. Using the calculated risk values at each state, we can then adapt the extent and frequency of monitoring accordingly. In states with higher degree of insufficiency, the medical staff may require to monitor more frequently and/or monitor more relevant measurements. These guidelines may be then encoded in the corresponding pathophysiological state. If integrated in the guidance system, such a model can provide risk driven sepsis screening and management.

4.4 Research Publications

Following is the published research paper on our initial findings on the use of state-based Bayesian networks to improve the efficiency of high-risk patient identification:

- Jiang, Yu, Lui Sha, **Maryam Rahmaniheris**, Binhua Wan, Mohammad Hosseini, Pengliu Tan, and Richard B. Berlin Jr. ”*Sepsis Patient Detection and Monitor Based on Auto-BN*” Journal of medical systems 40, no. 4 (2016): 1-10. [73]

Chapter 5

MD PnP: Device Configuration Safety

5.1 Organ-Centric Device Configuration Management

Medical systems need to inherently be run-time configurable. In acute care environments, the patient condition can change rapidly and this may result in a dynamic clinical environment. The safety and efficacy of patient care in such dynamic environment depend on continuously safe use of medical monitoring and therapeutic devices. For instance, $SpO2$ readings (saturation of peripheral oxygen) in a patient with circulatory shock are faulty and must be disregarded. Moreover, attaching the blood pressure cuff and oximeter on the same upper limb causes faulty $SpO2$ readings when the cuff is inflated. This is necessary since the use of faulty measurements prevents correct assessment of patient condition and endangers patient safety. Another example is a patient with acute lung injury, who requires a ventilator as part of her/his therapeutic intervention.

The examples mentioned above, represent changes in cardiovascular and pulmonary organ states.

Insufficinecy Category	**Organ Insufficinecy States**
Disturbed Blood Flow	Limb Ischemia, Cardiac Ischemia
Blood Pressure Dysregulation	Hypertension, Hypotension, Excessive & Maladaptive Fluctuation
Blood Gas Imbalance	Alkalosis, Acidosis, Hypercapnia, Hypocapnia, Hypoxemia
Cardiac Arrhythmia	Pulseless Ventricular Tachycardia (Pulseless V-Tach), Ventricular Fibrillation (V-Fib), Pulseless Electrical Activity (PEA), Asystole

Table 5.1: The Cardiovascular Organ Insufficiency States

As a result of such changes in the organ states, new devices may be plugged in or an existing device may be removed from clinical configuration. In fact, different organ states require different monitoring and therapeutic device configurations. Medicine is organized according to human organ systems. Each organ system performs a certain task in the body. Different diseases, can impair one or more of the physiological functions performed by the organ system. The pathophysiology of an organ system would describe how each of the functions are affected by different diseases. For example, the circulatory shock is a cardiovascular disease, which impairs the blood flow in the limbs. Impairment of the blood flow in the limbs is considered a cardiovascular insufficiency state while circulatory shock is a disease causing the insufficiency. Understanding the pathophysiology of an organ system is key to efficient management of monitoring and therapeutic medical devices. We propose a device configuration management approach that apply the proposed organ-centric disease patient model to systematically structure device configuration rules. Moreover, it is necessary to trigger the correct configuration requirements as the acute care environment changes. We have also developed the UPPAAL model of the organ-centric device configuration management system and verified a set of safety properties against the model.

We use the cardiac arrest case study to explain our approach. The cardiovascular system consists of the heart and blood vessels. The main responsibility of this organ system is to support blood circulation to provide nutrients, oxygen, etc. to different parts of the human body. Another task performed by this organ system is to support the $CO2$ and $O2$ exchanges in the lungs that stabilize blood PH, which is the measure of acidity of the blood. We can categorize the different pathophysiological states of the cardiovascular organ system into four main insufficiencies. The first category represent the conditions, in which the blood flow is blocked or partially blocked. This can be the result of different cardiovascular disease such as blood clot, hardening and narrowing of the artery, peripheral artery disorder or circulatory shock. The second category represents the abnormalities in heart rhythm, which leads to significant drop of cardiac output and impairs blood circulation. The insufficient blood circulation in the lungs to perform the $CO2$ and $O2$ exchange results in $CO2$ accumulation in the blood and causes a rise in blood PH level. This plays an important role in our case study on cardiac arrest. Abnormal blood gas levels and dysregulated blood pressure are the third and fourth categories. Table 5.1 shows a subset of cardiovascular states relevant to the intelligent management of devices. We have verified this abstract model of cardiovascular insufficiency states with the physician, who collaborates closely with our research group.

5.2 Monitoring and Therapeutic Configuration Safety

In this section, we consider a subset of device configuration safety constraints, commonly enforced in cardiac arrest scenario by medical staff. First, we describe the rules as they are taught and used clinically.

Conditional use of measurements and monitoring devices We describe two example rules of this type used in cardiac arrest scenario. For a patient with circulatory shock:

- $\mathbf{R_1}$: Oximeter and blood pressure cuff must not be used.
- $\mathbf{R_2}$: Central venous catheter must be used to measure blood pressure and oxygenation.

Inter-Device Interference Such interferences can be either direct or indirect. To illustrate the use of patient states, we focus on the indirect interferences that involves human physiology:

- $\mathbf{R_3}$: When the oximeter and blood pressure cuff are attached to the same upper limb the $SpO2$ readings, saturation of peripheral oxygen, must be discarded when the cuff is inflated.

Conditional use of therapeutic devices and intravenous medications Misuse of treatment devices and inappropriate administration of medications can lead to an unsafe state. For instance, for a cardiac arrest patient:

- $\mathbf{R_4}$: Defibrillator must not be used for non-shockable rhythms.
- $\mathbf{R_5}$: Epinephrine must be given to a patient with non-shockable rhythm.
- $\mathbf{R_6}$: Epinephrine is not effective for a cardiac arrest patient with low blood pH level. Bicarbonate must be given to the patient for Epinephrine to be effective.

Name	Physiology	Monitor Type	Current Reading
BP	Blood Pressure	Blood Pressure Cuff	80 mmHg
SpO2	Peripheral Oxygen Saturation	Oximeter	95%
Pulse	Pulse Rate	Oximeter	100 bpm
HR-EKG	Heart Rate	EKG Machine	95 bpm
CO2	Exhaled Carbon Dioxide	Capnometer	50%
CVP	Central Venous Pressure	Central Venous Catheter	80 mmHg
ScvO2	Central Venous Oxygen Saturation	Central Venous Catheter	80%
Heart-Rhythm	Heart Rhythm	EKG Machine	Ventricular Tachycardia
pH	Arterial Blood Acidity	Arterial Blood Gas Test	7.2

Table 5.2: Measurement Model and Example Instantiations

- $\mathbf{R_7}$: For the acidosis patient, calcium chloride ($CaCl2$) should not be administered in the same infusion or venous point as sodium bicarbonate ($NaHCO3$). Due to a chemical interaction, calcium will combine with sodium bicarbonate to form an insoluble precipitate [74].

It is important to note that the issue of potential side effects and adverse interaction of medications is quite complex and outside the scope of this paper [75]. *In this paper, we only consider a subset of safety requirements concerning medications, which ICU physicians ask our system to monitor at runtime.*
The minimum optimal set of actuators and measurements for a given patient state The example rules for cardiac arrest state are:

- $\mathbf{R_8}$: The optimal therapeutic configuration includes ventilator, infusion pump, and defibrillator.

- $\mathbf{R_9}$: The optimal set of physiological measurements include blood pressure, blood oxygenation, exhaled $CO2$ and heart electrical activity.

5.3 Modeling Device Configuration Safety

Now, we illustrate how these device and medication safety requirements can be organized more efficiently by the introduction of explicit organ states. Each requirement in the previous section describes a device or medication constraint which must be satisfied for a specific patient pathophysiological state.

Name	Monitor Type	Availability	Monitoring Status	Body Part	Measurement List
BPCuff	Blood Pressure Cuff	Available	In Use	Left Upper Limb	{BP}
Oximeter	Pulse Oximetery	Available	In Use	Right Upper Limb	{SpO2, Pulse}
cvCatheter	Central Venous Catheter	Available	Not In Use	Subclavian Vein	{ScvO2, CVP}
EKG	EKG Machine	Available	In Use	Chest	{Heart Rhythm}
ABG-Test	Arterial Blood Gas Test	Not Available	Not In Use	-	{pH, paCO2, paO2}

Table 5.3: Monitoring Device Model and Example Instantiations

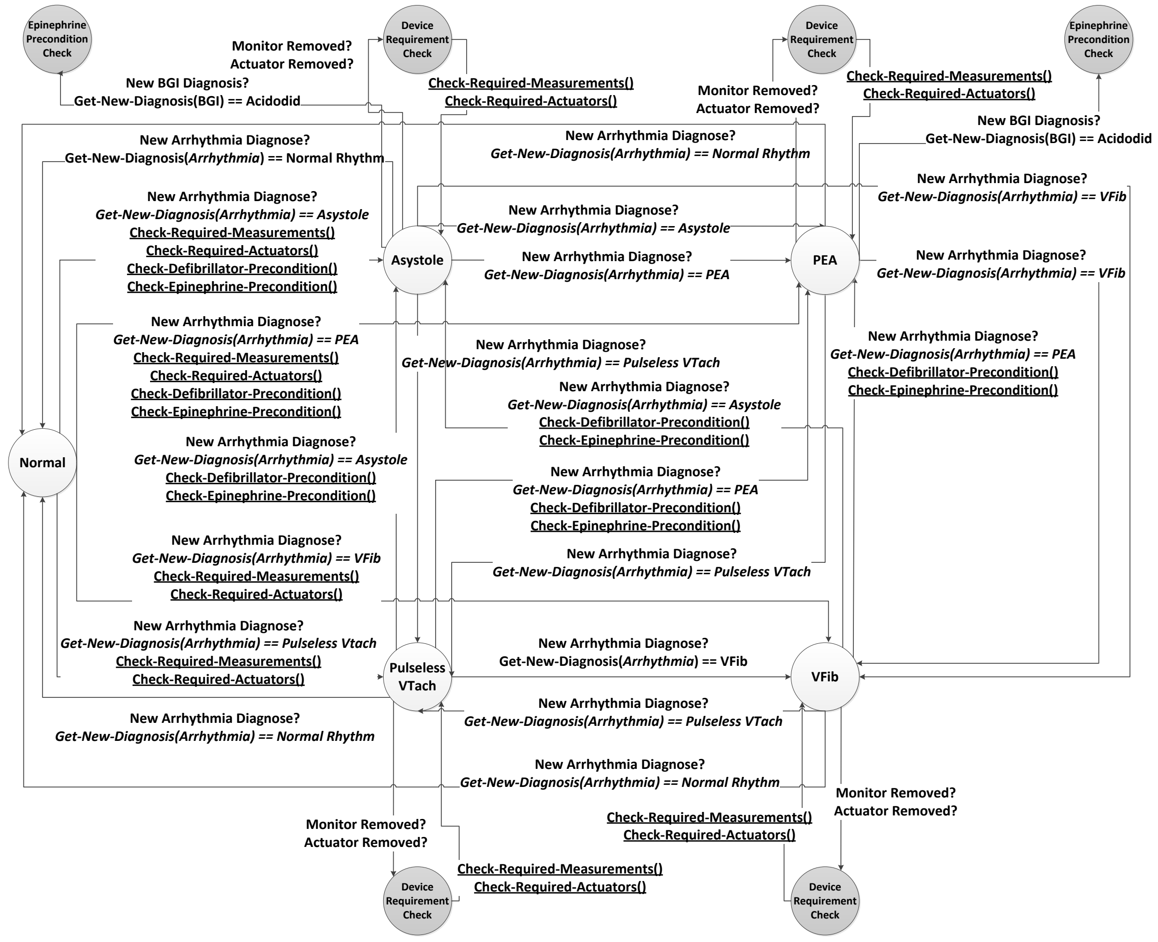

Epinephrine Precondition Check
Device Requirement Check
Device Requirement Check
Epinephrine Precondition Check
Monitor Removed?
Actuator Removed?
New BGI Diagnosis?
Get-New-Diagnosis(BGI) == Acidodid
Check-Required-Measurements()
Check-Required-Actuators()
Monitor Removed?
Actuator Removed?
Check-Required-Measurements()
Check-Required-Actuators()
New BGI Diagnosis?
Get-New-Diagnosis(BGI) == Acidodid
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Normal Rhythm
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Normal Rhythm
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == VFib
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Asystole
Check-Required-Measurements()
Check-Required-Actuators()
Check-Defibrillator-Precondition()
Check-Epinephrine-Precondition()
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Asystole
Asystole
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == PEA
PEA
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == VFib
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == PEA
Check-Required-Measurements()
Check-Required-Actuators()
Check-Defibrillator-Precondition()
Check-Epinephrine-Precondition()
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Pulseless VTach
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == PEA
Check-Defibrillator-Precondition()
Check-Epinephrine-Precondition()
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Asystole
Check-Defibrillator-Precondition()
Check-Epinephrine-Precondition()
Normal
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Asystole
Check-Defibrillator-Precondition()
Check-Epinephrine-Precondition()
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == PEA
Check-Defibrillator-Precondition()
Check-Epinephrine-Precondition()
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == VFib
Check-Required-Measurements()
Check-Required-Actuators()
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Pulseless VTach
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Pulseless Vtach
Check-Required-Measurements()
Check-Required-Actuators()
Pulseless VTach
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == VFib
VFib
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Normal Rhythm
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Pulseless VTach
New Arrhythmia Diagnose?
Get-New-Diagnosis(Arrhythmia) == Normal Rhythm
Monitor Removed?
Actuator Removed?
Check-Required-Measurements()
Check-Required-Actuators()
Monitor Removed?
Actuator Removed?
Check-Required-Measurements()
Check-Required-Actuators()
Device Requirement Check
Device Requirement Check

Figure 5.1: Arrhythmia State Machine Focused on Cardiac Arrest Scenario

The patient state can be represented by a combination of organ states. We call the first element of each requirement ”*organ state predicate*” and the second element ”*configuration predicate*”.

In this paper, we assume each device configuration for a common patient state is standardized based on generally accepted medical principles. A configuration for a given patient state is the combination

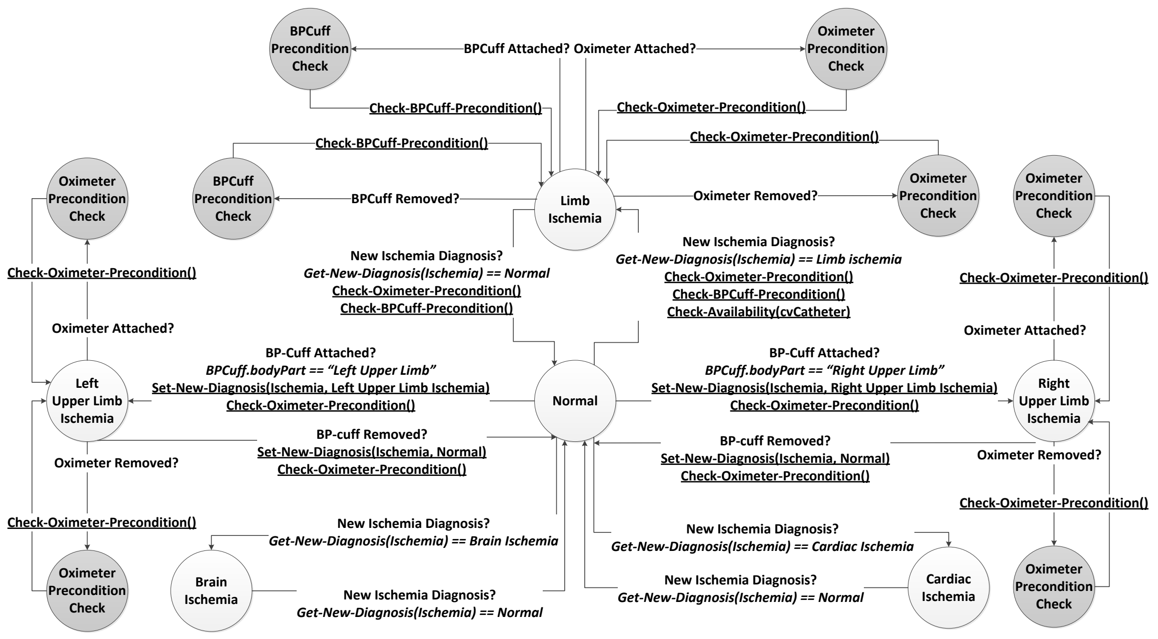

BPCuff Precondition Check
BPCuff Attached? Oximeter Attached?
Oximeter Precondition Check
Check-BPCuff-Precondition()
Check-Oximeter-Precondition()
Check-BPCuff-Precondition()
Check-Oximeter-Precondition()
Oximeter Precondition Check
BPCuff Precondition Check
BPCuff Removed?
Limb Ischemia
Oximeter Removed?
Oximeter Precondition Check
Oximeter Precondition Check
New Ischemia Diagnosis?
Get-New-Diagnosis(Ischemia) == Normal
Check-Oximeter-Precondition()
Check-BPCuff-Precondition()
New Ischemia Diagnosis?
Get-New-Diagnosis(Ischemia) == Limb ischemia
Check-Oximeter-Precondition()
Check-BPCuff-Precondition()
Check-Availability(cvCatheter)
Check-Oximeter-Precondition()
Check-Oximeter-Precondition()
Oximeter Attached?
Oximeter Attached?
BP-Cuff Attached?
BPCuff.bodyPart == "Left Upper Limb"
Set-New-Diagnosis(Ischemia, Left Upper Limb Ischemia)
Check-Oximeter-Precondition()
BP-Cuff Attached?
BPCuff.bodyPart == "Right Upper Limb"
Set-New-Diagnosis(Ischemia, Right Upper Limb Ischemia)
Check-Oximeter-Precondition()
Left Upper Limb Ischemia
Normal
Right Upper Limb Ischemia
BP-cuff Removed?
Set-New-Diagnosis(Ischemia, Normal)
Check-Oximeter-Precondition()
BP-cuff Removed?
Set-New-Diagnosis(Ischemia, Normal)
Check-Oximeter-Precondition()
Oximeter Removed?
Oximeter Removed?
Check-Oximeter-Precondition()
Check-Oximeter-Precondition()
New Ischemia Diagnosis?
Get-New-Diagnosis(Ischemia) == Brain Ischemia
New Ischemia Diagnosis?
Get-New-Diagnosis(Ischemia) == Cardiac Ischemia
Oximeter Precondition Check
Brain Ischemia
New Ischemia Diagnosis?
Get-New-Diagnosis(Ischemia) == Normal
New Ischemia Diagnosis?
Get-New-Diagnosis(Ischemia) == Normal
Cardiac Ischemia
Oximeter Precondition Check

Figure 5.2: Ischemia State Machine

of monitoring and therapeutic configurations. The monitoring configuration refers to the set of all commonly used monitoring devices and their corresponding attributes. The therapeutic configuration represents the set of medical actuators and IV (intravenous) medications for a given patient state. We use a unified model for medical devices, physiological measurements, and IV medications. Each of the specific fields or attributes of the elements in the model are accessed and used to describe and evaluate the safety constraints. For instance, the monitoring device model includes the *body part* where the device is attached to, and the *monitoring status* indicating whether the measurements generated by the device are being used. Tables 5.2 and 5.3 show examples of measurement and monitoring device model instantiations. Actuator and medication model instantiations are not presented due to space limitation; however, most of the chosen attributes for these models are self-describing.

Each of the main categories of organ state insufficiencies is realized in an individual state machine. Figures 5.1, 5.2, and 5.3 show the state machines of *cardiac arrhythmia*, *ischemia*, and *blood gas imbalance (BGI)* categories, respectively. Each machine is a labeled transition system. The nodes in each machine represent different organ states in the corresponding category. The gray circles represent the states, in which the configuration safety is re-evaluated. The clinical events, such as a new diagnosis,

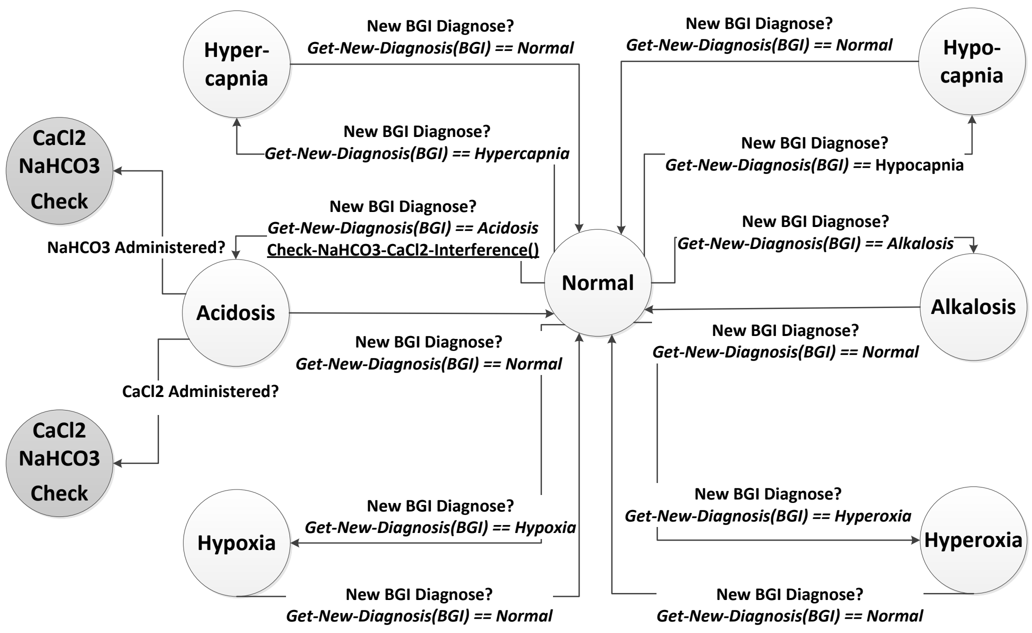

New BGI Diagnose?
Get-New-Diagnosis(BGI) == Normal
Hyper-capnia
Hypo-capnia
New BGI Diagnose?
Get-New-Diagnosis(BGI) == Hypercapnia
New BGI Diagnose?
Get-New-Diagnosis(BGI) == Hypocapnia
CaCl2 NaHCO3 Check
New BGI Diagnose?
Get-New-Diagnosis(BGI) == Acidosis
Check-NaHCO3-CaCl2-Interference()
NaHCO3 Administered?
New BGI Diagnose?
Get-New-Diagnosis(BGI) == Alkalosis
Normal
Acidosis
Alkalosis
CaCl2 Administered?
Hypoxia
Hyperoxia
New BGI Diagnose?
Get-New-Diagnosis(BGI) == Hypoxia
New BGI Diagnose?
Get-New-Diagnosis(BGI) == Hyperoxia

Figure 5.3: Gas Imbalance State Machine

change of a device status or administration of a new medication, trigger the transition to these states.

The transition between the states are labeled with guards and actions. A guard is a boolean expression, and the transition labeled with a guard can be enabled when the guard evaluates to true. The action must be performed when the transition is enabled. The transition links may also be labeled with communication actions. In state machine language, $c?$ and $c!$ represent the send and receipt actions on the communication channel c, respectively. For example, in our system, the transition between different organ states happens according to the physician diagnosis. For example, when the patient is diagnosed with acidosis, the physician interface machine communicates this information by sending *New BGI Diagnosis!* to *Gas Imbalance* state machine. The receipt of this diagnosis information is indicated by the outgoing edges from Normal state in figure 5.3, which are labeled with *New BGI Diagnosis?* communication action.

As mentioned above, the device configuration safety requirements often depend on the organ insufficiency states rather than diseases. Therefore, the detailed disease information can be encapsulated. For example, when physician diagnoses the patient with circulatory shock, the disease is abstracted away in the ischemia state model and is hidden from the rest of organ insufficiency machines. This is done by defining *Get-New-Diagnosis(Ischemia)* function. This function checks the value of *Ischemia-Diagnosis* set by the physician. If the diagnosis refers to any of the diseases causing limb ischemia, such as circulatory shock, it returns *Limb Ischemia*. The information regarding the relation between diseases and organ insufficiencies must come from the physician. We only encode the information given by the physician.

5.3.1 Encoding Safety Checks in Organ Insufficiency Machines

At each point during system operation, only a subset of configuration safety requirements must be monitored. These are the requirements, which are encoded in the state(s) representing the current patient condition. For example, for a cardiac arrest patient with asystole rhythm and low blood pH, cardiac dysrhythmia and blood gas imbalance state machines would be in *Asystole* and *Acidosis* states, respectively. The requirements $\mathbf{R_4}$, $\mathbf{R_5}$, $\mathbf{R_6}$, $\mathbf{R_8}$, and $\mathbf{R_9}$ must be encoded in *Asystole* state and $\mathbf{R_7}$ must be encoded in *Acidosis* state. These requirement would be the only requirements to monitor for the given patient condition.

The requirement $\mathbf{R_6}$ contains two different patient states. Since cardiac arrest is the primary condition we encode this requirement in the corresponding states in arrhythmia machine. However, we must add an interaction point between arrhythmia and *BGI* machines at the appropriate states. This is explained in the next subsection.

Now, we take a closer look at the requirements we need to encode in different organ states for the cardiac arrest scenario. Let's consider the patient condition in $\mathbf{R_1}$ and $\mathbf{R_2}$ that describes a disease, in which the blood flow to peripherals are disturbed. As mentioned above, this is a cardiovascular insufficiency state known as limb ischemia. This state interferes with the correct operation of blood pressure cuff and oximeter, which depends on non-disturbed blood flow in the upper limbs. Therefore, we encode the requirements on correct operation of blood pressure cuff and pulse oximetery in *Limb Ischemia* state in *Ischemia* state machine. This is encoded by adding two functions *Check-Oximeter-Precondition()* and *Check-BPCuff-Precondition()* to the action item of the transition to Limb Ischemia state.

For cardiovascular conditions, blood pressure and oxygenation must be monitored continuously since they are basic indicators of cardiovascular functions in the body. Therefore, when patient state indicates limb ischemia monitoring devices other than oximeter and blood pressure cuff, such as central venous catheter, must be used to measure blood pressure and oxygen saturation. This is encoded by defining the third function checking the availability of *Check-Availability(cvCatheter)*. These functions would also be used to re-evaluate $\mathbf{R_1}$ and $\mathbf{R_2}$ when any event relevant to these requirements happen.

The blood flow can also be disturbed by a medical device. The blood pressure cuff disturbs the blood flow on the upper limb when the cuff is inflated. Therefore, according to $\mathbf{R_3}$, if the oximeter is attached to the same upper limb as the cuff is attached, the organ state precondition for validity of blood oxygenation, measured by the oximeter, is violated. These are encoded in *Right Upper Limb Ischemia* and *Left Upper Limb Ischemia* states in *Ischemia* state machine. We define *Check-Oximeter-Precondition()* function to check for any potential interference. This is the only case, in which we transition to a new organ state without any signal from physician. When blood pressure cuff is in use and we are in *Normal* state we transition to one of *Upper Limb Ischemia* states depending on the body part, to which the cuff is attached.

The requirement $\mathbf{R_4} - \mathbf{R_6}$ is encoded in *PEA* and *Asystole* states and is checked using *Check-Defibrillator-Precondition()* function. In our model, the requirement $\mathbf{R_7}$ is encoded in *Acidosis* state in *BGI* machine. Although this requirement is related to a chemical reaction of two medication independent of patient state, in our cardiac arrest scenario we choose to encode it in a state, where one or both of these medications are potentially used as part of treating the corresponding organ insufficiency. This follows our unified model while ensuring the safety requirement is encoded properly. We define function *Check-NaHCO3-CaCl2-Interference()* to check this requirements whenever needed.

We define *Check-Measurement-Configuration()* and *Check-Actuator-Configuration()* functions to check $\mathbf{R_8}$ and $\mathbf{R_9}$. These requirements must be encoded in all of the four cardiac arrest states: *Asystole*, *PEA*, *Pulseless VTach*, and *VFib*. For the actuators, we only require their availability. The actuating status is adapted dynamically according to organ states. The rules describing the required measurements do not specify a monitoring device.The two functions added to the action element of the transitions from *Normal* state to any of these four states. When we are in one of these states, these requirements would be re-evaluated when there is a change in device configuration.

5.3.2 Modeling the Interactive Safety Requirements

The next logical step is to identify and model any dependencies or interactions between these set of requirements. Note that the there is a common element in the configuration constrain of $\mathbf{R_5}$ and $\mathbf{R_6}$, which is the medication epinephrine. According to AHA, "high-dose epinephrine may improve coronary perfusion and increase vascular resistance to promote initial return of spontaneous circulation during CPR" [76]. However, epinephrine can also lead to transient respiratory acidosis and decrease in blood pH level [77]. In a patient with already low blood pH, this effect can be exacerbated.

To model such interactive or potentially conflicting requirements in state machine language, we add a ***synchronization point*** to the involved states. In this case, the interaction between $\mathbf{R_5}$ and $\mathbf{R_6}$, as described above, is encoded in non-shockable cardiac arrest state which include *PEA* and *Asystole*. Therefore, we add a ***synchronization point*** to *PEA* and *Asystole* states. When a new diagnosis for *BGI* is received, which indicated acidosis, this requirements is re-evaluated by calling the function *Check-Epinephrine-Precondition().*

Defining these synchronization points results in a well-defined interface between different organ state machines and limits the their interactions to these predefined points. However, we should note that there might be other type of dependency relations between medical rules, which may require a different choice of modeling. This is a topic of future work.

Having the organ insufficiency state machines for the cardiac arrest case with encoded safety requirements in the relevant states, configuration safety can be validated as the environment and patient condition changes. The definition of safety check functions described above are shown in pseudocodes 2, 3, 4 and 5. The global variable $\mathbf{R_i}$-*Violated* indicates whether the corresponding requirement is violated or not. The definition of some of the simpler functions are not shown due to space limitations.

5.4 Formal Verification of Device Configuration Safety Model

We follow the approach explained in section 5.3 to encode the rules and safety evaluation algorithm in each organ state automaton. The only difference is that the organ state automaton immediately transition back from the violation states after updating the appropriate $Violation$ variables. This is to ensure that the automata are in the states, which can synchronously receive the configuration and

Pseudocode 2 Check-Required-Measurements(Diagnosis)

```

Boolean $Oxygen - Available$ = FALSE;
Boolean $BP - Available$ = FALSE;
Boolean $Heart - Rhythm - Available$ = FALSE;
$\mathbf{R_9 - Violated}$ = FALSE;

IF $Diagnosis \in Arrhythmia$ THEN
    IF ($\exists$ monitor m $where$ (($m.Availability == Available$) AND ($m.MonitoringStatus == InUse$) AND ($SpO2 \in$
     $m.MeasurementList$ OR $ScvO2 \in m.MeasurementList$))) THEN
        $Oxygen - Available$ = TRUE;
    END IF;
    IF ($\exists$ monitor m $where$ (($m.Availability == Available$) AND ($m.MonitoringStatus == InUse$) AND ($BP \in$
     $m.MeasurementList$ OR $CVP \in m.MeasurementList$))) THEN
        $BP - Available$ = TRUE;
    END IF;
    IF ($\exists$ monitor m $where$ (($m.Availability == Available$) AND ($m.MonitoringStatus == InUse$) AND (
     $Heart - Rhythm \in m.MeasurementList$)))THEN
        $Heart - Rhythm - Available$ = TRUE;
    END IF;
    IF (($Oxygen - Available$ == TRUE) AND ($BP - Available$ == TRUE) AND
($Heart - Rhythm - Available$ == TRUE)) THEN
        $\mathbf{R_9 - Violated}$ = FALSE;
    END IF;
END IF;
```

patient state change signals and immediately evaluate the current enforced rules and elect the new set of rules to be enforced.

As mentioned in the previous section, the transition between different organ states are enabled by the physician's diagnosis. Figures 5.6 show the physician automata which is responsible for sending the $Diagnosed?$ and $Resolved?$ synchronization signals for different organ states automata. We show the oximeter and blood pressure cuff automata, as examples of device automata, in figure 5.7. Notice, we have defined broadcast synchronization channels, whose senders are the devices. This ensures that any change in device configuration is communicated synchronously to other organ state automata. Similar channels are defined in the infusion pump automaton for different intravenous medications.

Pseudocode 3 Check-Oximeter-Precondition()

```

$\mathbf{R_3 - Violated}$ = FALSE;
IF $Oximeter.BodyPart$ == $BPCuff.BodyPart$ THEN
    $\mathbf{R_3 - Violated}$ = TRUE;
END IF;
IF $(Get - New - Diagnosis(Ischemia)$ == $LimbIschemia)$ THEN
    $\mathbf{R_3 - Violated}$ = TRUE;
END IF;
IF $((Get - New - Diagnosis(Ischemia)$ == $RightUpperLimbIschemia)$ AND $(Oximeter.BodyPart$ ==
    $RightUpperLimb))$ THEN
    $\mathbf{R_3 - Violated}$ = TRUE;
END IF;
IF $((Get - New - Diagnosis(Ischemia)$ == $LeftUpperLimbIschemia)$ AND $(Oximeter.BodyPart$ ==
    $LeftUpperLimb))$ THEN
    $\mathbf{R_3 - Violated}$ = TRUE;
END IF;
```

Pseudocode 4 Check-Epinephrine-Precondition()

```
$\mathbf{R_5 - Violated}$ = FALSE;
$\mathbf{R_6 - Violated}$ = FALSE;
IF $((Get - New - Diagnosis(BGI)$ == $Normal)$ AND $(Epi.AdministerationStatus$ == $NotAdminstered))$ THEN
        $\mathbf{R_5 - Violated}$ = TRUE;
END IF;
IF $((Get - New - Diagnosis(BGI)$ == $Acidosis)$ AND $(Epi.AdministerationStatus$ == $Adminstered))$ THEN
    $\mathbf{R_6 - Violated}$ = TRUE;
END IF;
```

Pseudocode 5 Check-NaHCO3-CaCl2-Interference()

```
$\mathbf{R_7 - Violated}$ = FALSE;
IF $((NaHCO3.AdministerationStatus$ == $Adminstered)$ AND $(CaCl2.AdministerationStatus$ == $Adminstered)$
    AND $(NaHCO3.InfusionPump$ == $CaCl2.InfusionPump))$ THEN
    $\mathbf{R_7 - Violated}$ = TRUE;
END IF;
```

The ischemia model is presented in figures 5.2. Notice ***Rules 1-3*** are encoded in limb ischemia state. The ischemia automata communicates with other organ states including cardiac dysrhythmia

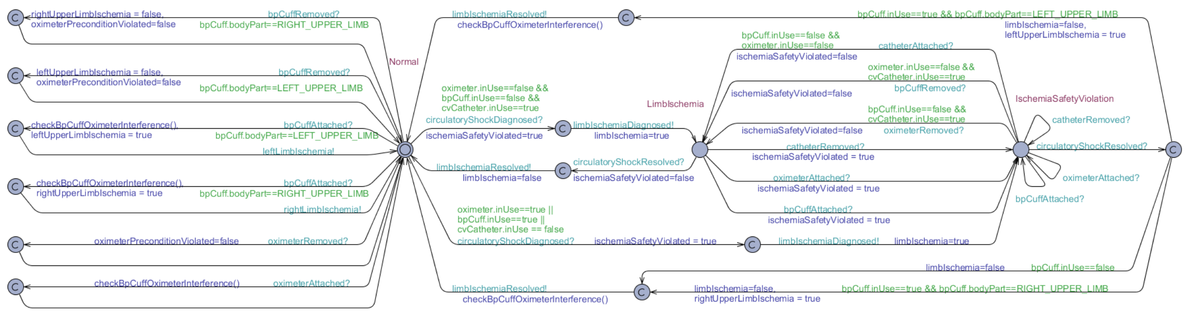

rightUpperLimbIschemia = false,
oximeterPreconditionViolated=false
bpCuffRemoved?
bpCuff.bodyPart==RIGHT_UPPER_LIMB
leftUpperLimbIschemia = false,
oximeterPreconditionViolated=false
bpCuffRemoved?
bpCuff.bodyPart==LEFT_UPPER_LIMB
checkBpCuffOximeterInterference(),
leftUpperLimbIschemia = true
bpCuffAttached?
bpCuff.bodyPart==LEFT_UPPER_LIMB
leftLimbIschemia!
checkBpCuffOximeterInterference(),
rightUpperLimbIschemia = true
bpCuffAttached?
bpCuff.bodyPart==RIGHT_UPPER_LIMB
rightLimbIschemia!
oximeterPreconditionViolated=false
oximeterRemoved?
checkBpCuffOximeterInterference()
oximeterAttached?
Normal
limbIschemiaResolved!
checkBpCuffOximeterInterference()
oximeter.inUse==false &&
bpCuff.inUse==false &&
cvCatheter.inUse==true
circulatoryShockDiagnosed?
ischemiaSafetyViolated=true
limbIschemiaDiagnosed!
limbIschemia=true
LimbIschemia
limbIschemiaResolved!
limbIschemia=false
circulatoryShockResolved?
ischemiaSafetyViolated=false
oximeter.inUse==true ||
bpCuff.inUse==true ||
cvCatheter.inUse == false
circulatoryShockDiagnosed?
ischemiaSafetyViolated = true
limbIschemiaDiagnosed!
limbIschemia=true
limbIschemiaResolved!
checkBpCuffOximeterInterference()
limbIschemia=false,
rightUpperLimbIschemia = true
bpCuff.inUse==true && bpCuff.bodyPart==RIGHT_UPPER_LIMB
limbIschemia=false
bpCuff.inUse==false
bpCuff.inUse==true && bpCuff.bodyPart==LEFT_UPPER_LIMB
limbIschemia=false,
leftUpperLimbIschemia = true
bpCuff.inUse==false &&
oximeter.inUse==false
ischemiaSafetyViolated=false
catheterAttached?
oximeter.inUse==false &&
cvCatheter.inUse==true
bpCuffRemoved?
ischemiaSafetyViolated=false
bpCuff.inUse==false &&
cvCatheter.inUse==true
ischemiaSafetyViolated=false
oximeterRemoved?
catheterRemoved?
ischemiaSafetyViolated = true
oximeterAttached?
ischemiaSafetyViolated = true
bpCuffAttached?
ischemiaSafetyViolated = true
IschemiaSafetyViolation
catheterRemoved?
circulatoryShockResolved?
oximeterAttached?
bpCuffAttached?

Figure 5.4: Ischemia Automaton

using $ischemiaDiagnosed$ and $ischemiaResolved$ channels. The interface between the ischemia state and other organ states are well-defined and is limited to only ischemia diagnosis and resolution signals. Special cases of upper limb ischemia caused by blood pressure cuff is also modeled in ischemia automata.

The cardiac dysrhythmia automaton is shown in 5.5. The ***Rules 4, 6*** are encoded in the non-shockable cardiac arrest state and ***Rule 5*** is encoded in shockable cardiac arrest state. ***Rules 8, 9*** are encoded in all cardiac arrest states. When the automaton transitions to a different state the enforced set of rules are updated automatically since the rules are encoded in each state. For example, when the non-shockable cardiac arrest state is entered the rule preventing the activation of defibrillator is added automatically since ***Rule 4*** is encoded in this state. When an organ state encoded in a different automaton is diagnosed or resolved the corresponding signal is communicated to cardiac dysrhythmia automaton which triggers the re-evaluation of the currently enforced rules. For example in any of the cardiac arrest states, receiving $limbIscemiaResolved?$ results in re-evaluation of ***Rules 9.a, 9.b*** and checking the availability of oximeter and blood pressure cuff. The changes in device configuration are communicated synchronously as well using the corresponding broadcast channels. ***Rules 7*** is encoded in the acidosis state in the blood gas imbalance automaton (not shown due to space limitation). Next, we describe several examples of properties verified by UPPAAL tool.

5.4.1 Properties

We verified the absence of deadlock and reachability of all the states in the model. In addition, a set of properties, indicating that any violation of the defined safety rules are always detected, has been also

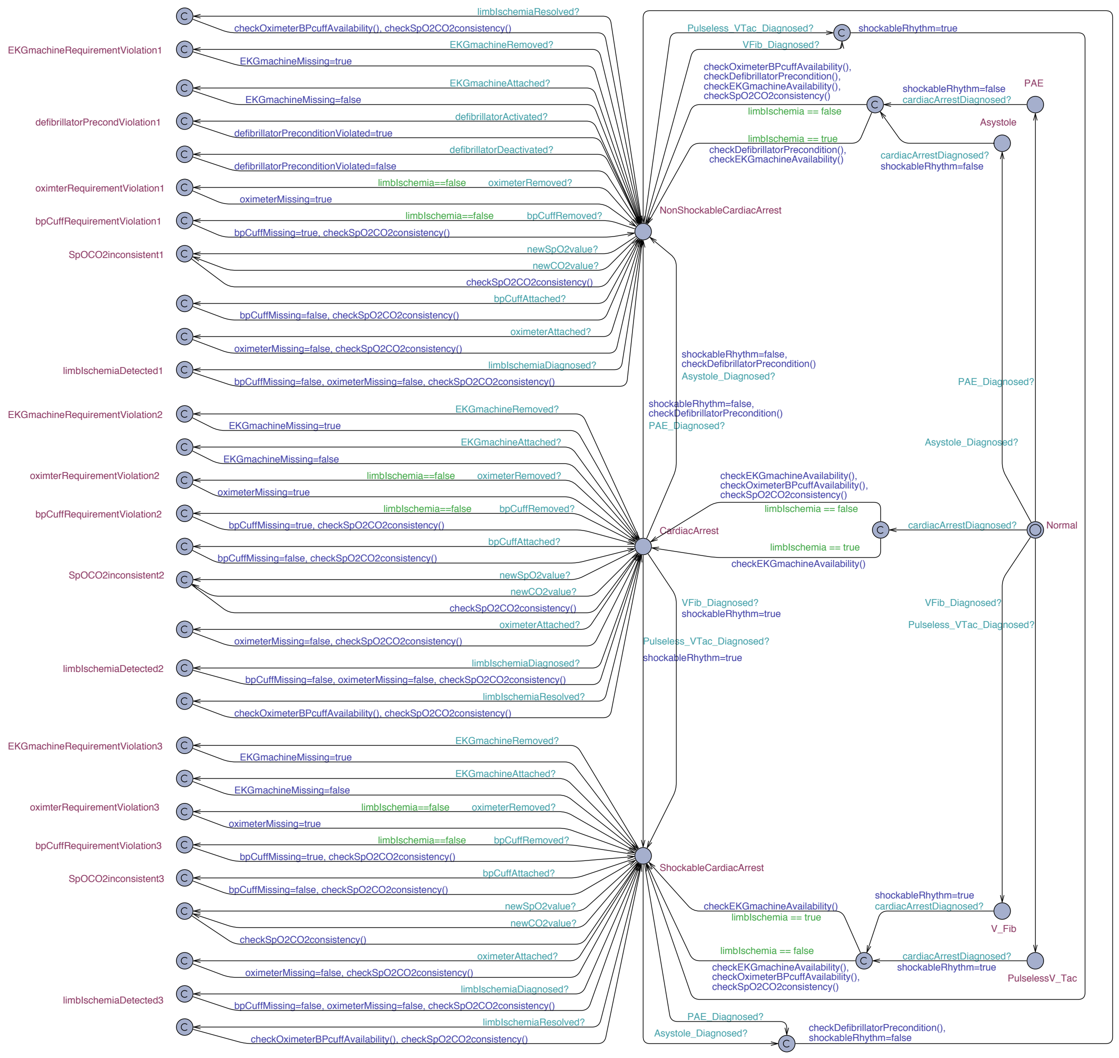

limbIschemiaResolved?
checkOximeterBPcuffAvailability(), checkSpO2CO2consistency()
EKGmachineRequirementViolation1
EKGmachineRemoved?
EKGmachineMissing=true
EKGmachineAttached?
EKGmachineMissing=false
defibrillatorPrecondViolation1
defibrillatorActivated?
defibrillatorPreconditionViolated=true
defibrillatorDeactivated?
defibrillatorPreconditionViolated=false
oximterRequirementViolation1
limbIschemia==false
oximeterRemoved?
oximeterMissing=true
bpCuffRequirementViolation1
limbIschemia==false
bpCuffRemoved?
bpCuffMissing=true, checkSpO2CO2consistency()
SpOCO2inconsistent1
newSpO2value?
newCO2value?
checkSpO2CO2consistency()
bpCuffAttached?
bpCuffMissing=false, checkSpO2CO2consistency()
oximeterAttached?
oximeterMissing=false, checkSpO2CO2consistency()
limbIschemiaDetected1
limbIschemiaDiagnosed?
bpCuffMissing=false, oximeterMissing=false, checkSpO2CO2consistency()
EKGmachineRequirementViolation2
EKGmachineRemoved?
EKGmachineMissing=true
EKGmachineAttached?
EKGmachineMissing=false
oximterRequirementViolation2
limbIschemia==false
oximeterRemoved?
oximeterMissing=true
bpCuffRequirementViolation2
limbIschemia==false
bpCuffRemoved?
bpCuffMissing=true, checkSpO2CO2consistency()
bpCuffAttached?
bpCuffMissing=false, checkSpO2CO2consistency()
SpOCO2inconsistent2
newSpO2value?
newCO2value?
checkSpO2CO2consistency()
oximeterAttached?
oximeterMissing=false, checkSpO2CO2consistency()
limbIschemiaDetected2
limbIschemiaDiagnosed?
bpCuffMissing=false, oximeterMissing=false, checkSpO2CO2consistency()
limbIschemiaResolved?
checkOximeterBPcuffAvailability(), checkSpO2CO2consistency()
EKGmachineRequirementViolation3
EKGmachineRemoved?
EKGmachineMissing=true
EKGmachineAttached?
EKGmachineMissing=false
oximterRequirementViolation3
limbIschemia==false
oximeterRemoved?
oximeterMissing=true
bpCuffRequirementViolation3
limbIschemia==false
bpCuffRemoved?
bpCuffMissing=true, checkSpO2CO2consistency()
SpOCO2inconsistent3
bpCuffAttached?
bpCuffMissing=false, checkSpO2CO2consistency()
newSpO2value?
newCO2value?
checkSpO2CO2consistency()
oximeterAttached?
oximeterMissing=false, checkSpO2CO2consistency()
limbIschemiaDetected3
limbIschemiaDiagnosed?
bpCuffMissing=false, oximeterMissing=false, checkSpO2CO2consistency()
limbIschemiaResolved?
checkOximeterBPcuffAvailability(), checkSpO2CO2consistency()
Pulseless_VTac_Diagnosed?
shockableRhythm=true
VFib_Diagnosed?
checkOximeterBPcuffAvailability(), checkDefibrillatorPrecondition(), checkEKGmachineAvailability(), checkSpO2CO2consistency()
limbIschemia == false
shockableRhythm=false
cardiacArrestDiagnosed?
PAE
Asystole
limbIschemia == true
checkDefibrillatorPrecondition(), checkEKGmachineAvailability()
cardiacArrestDiagnosed?
shockableRhythm=false
NonShockableCardiacArrest
shockableRhythm=false, checkDefibrillatorPrecondition()
Asystole_Diagnosed?
shockableRhythm=false, checkDefibrillatorPrecondition()
PAE_Diagnosed?
PAE_Diagnosed?
Asystole_Diagnosed?
checkEKGmachineAvailability(), checkOximeterBPcuffAvailability(), checkSpO2CO2consistency()
limbIschemia == false
CardiacArrest
cardiacArrestDiagnosed?
Normal
limbIschemia == true
checkEKGmachineAvailability()
VFib_Diagnosed?
shockableRhythm=true
Pulseless_VTac_Diagnosed?
shockableRhythm=true
VFib_Diagnosed?
Pulseless_VTac_Diagnosed?
ShockableCardiacArrest
checkEKGmachineAvailability()
limbIschemia == true
shockableRhythm=true
cardiacArrestDiagnosed?
V_Fib
limbIschemia == false
checkEKGmachineAvailability(), checkOximeterBPcuffAvailability(), checkSpO2CO2consistency()
cardiacArrestDiagnosed?
shockableRhythm=true
PulselessV_Tac
PAE_Diagnosed?
Asystole_Diagnosed?
checkDefibrillatorPrecondition(), shockableRhythm=false

Figure 5.5: Cardiac Dysrhythmia Automaton

(not cvCatheter.inUse and limbIschemia) $--->$ ischemiaSafetyViolated
(oximeter.inUse and limbIschemia) $--->$ ischemiaSafetyViolated
(bpCuff.inUse and limbIschemia) $--->$ ischemiaSafetyViolated
(bpCuff.inUse and oximeter.inUse and (oximeter.bodyPart == bpCuff.bodyPart)) $--->$ oximeterPreconditionViolated

Table 5.4: Examples of Verified Safety Properties

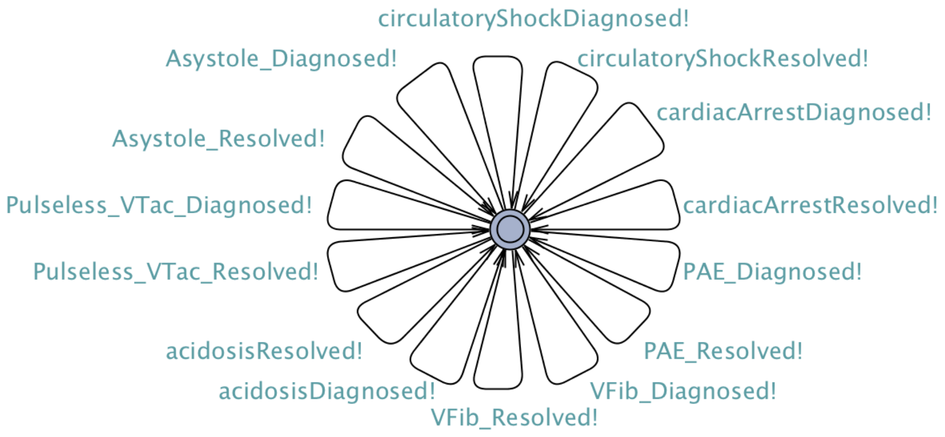


Figure 5.6: Physician Automaton

verified. A set of example properties are presented in table 5.4. The properties are of the form **leads to** or in other words **response** properties written as $\alpha --> \beta$ in UPPAAL tool. This properties are read as whenever α is satisfied, then eventually β will be satisfied [78]. As can be seen from table 5.4, the proposed rule structure and evaluation algorithm ensure that violation of any of the defined safety rules are always detected.

We should also note that in [79], we proposed a well-defined interface between different organ automata and limited the their interactions to a set of predefined points. *However, this can decrease the modularity of our organ-based modeling approach; if there is the need for a modification of an organ model, all the other organ models, which have any interaction with the modified model, must be modified as well*. Therefore, in [80], we move to a more modular approach by developing a central manager and encapsulating the interaction between different organ automata within the central manager. The models presented in this chapters are developed according to the modular approach.

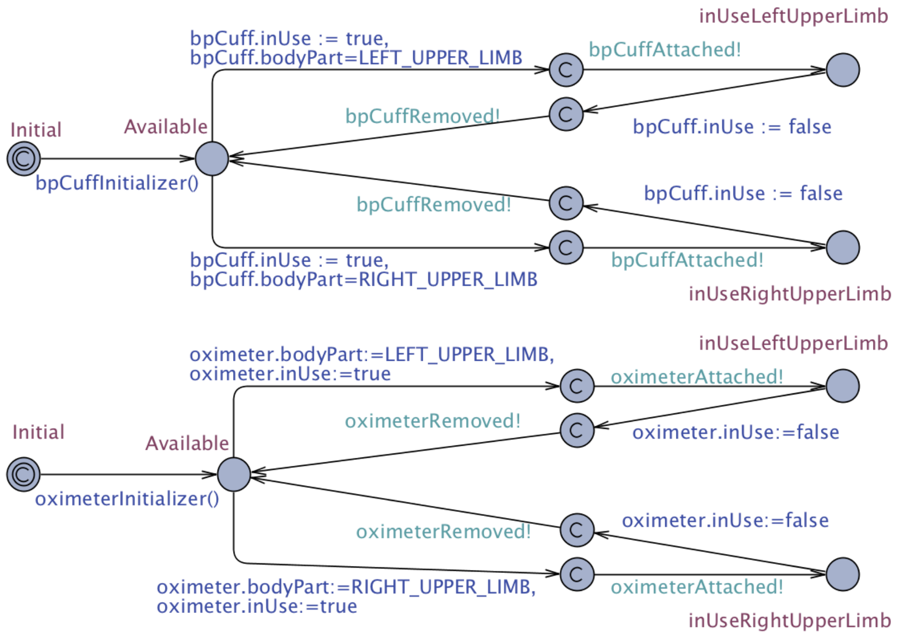


Figure 5.7: Pulse Oximeter and Blood Pressure Cuff Automata

5.5 Research Publications

The following lists our publications on dynamic device configuration safety and the use of organ-centric paradigm to model and ensure run-time safety of medical systems:

- **M. Rahmaniheris**, L. Sha, R. B., Berlin, and J. M. Goldman, "*Towards a Cyber-medical Model for Device Configuration Safety in Acute Care*", In 2014 IEEE Healthcare Innovation Conference (HIC), October 2014 (pp. 118-124) [81]

- **M. Rahmaniheris**, W. Kang, L. Lee, L. Sha, R. B. Berlin, and J. M. Goldman, "*Modeling and Architecture Design of an MDPnP Acute Care Monitoring Systems*", The 26th IEEE International Symposium on Computer-Based Medical Systems (CBMS), June 2013 (Demo paper) [82]

- W. Kang, P. Wu, **M. Rahmaniheris**, L. Sha, R. B. Berlin, J. M. Goldman, "*Towards Organ-Centric Compositional Development of Safe Networked Supervisory Medical Systems*", The 26th IEEE International Symposium on Computer-Based Medical Systems (CBMS), June 2013 [83]

Chapter 6

MD PnP: A Framework for the Safe Analysis

One goal of the Medical Device Plug-and-Play (MD PnP) program is the adoption of medical device interoperability to significantly mitigate preventable medical errors by developing the technological foundation to design the next generation medical systems. The ”Patient-Centric Integrated Clinical Environment” (ICE) standard is one such development, which is described in ASTM 2761-09(2013) [50]. This standard describes the logical elements of an MD PnP system but does not specify implementation details.

A MD PnP system supports a range of safe medical system configurations, where medical devices are ”plugged in” as needed, information from different sources are integrated, and medical actions are coordinated to provide an integrated clinical environment. An ICE typically consists of an ICE supervisor, an ICE network controller, ICE device interface and etc. The MD PnP system described in this report follows a similar structure and therefore MD PnP and ICE may be used interchangeably when describing different components of an integrated medical system.

To facilitate the development of medical device interoperability, the MD PnP program has developed an open source implementation of the Integrated Clinical Environment and made it freely available on SourceForge [51]. The platform consists of software device adapters for medical devices (including anesthesia machines, ventilators, and patient monitors), Object Management Group (OMG) Data Distribution Service (DDS) for Real-Time systems standard middle-ware, and demonstration applications. The choice of the MD PnP platform, namely, the computers, OS, and network, is up to the user. There are no approved platforms by any agency that support plug and play. This is because plug and play could generate new configurations not considered during the approval process. Unverified new configurations of platform may or may not work, hence, the need for assurance of life cycle attributes. Furthermore, all existing commercial platforms include a disclaimer indicating that these platforms are not designed to be used for safety critical systems and applications.

This document provides guidelines to assist users to safely use an MD PnP system consisting of a commercial platform not designed for safety critical applications. The MD PnP system is said to be safe for an intended medical application, if none of the MD PnP components can become a causal factor of the known safety hazards when the medical devices are functioning according to their specifications.

In the next section, we present an overview of the MD PnP system components and the safety requirements for each component. Before discussing safety analysis, a brief introduction on fault trees is provided in Section 6.2. In Section 6.3, we present the safety analysis procedure used in this report; two different approaches to ensure open-loop safety[1] of the MD PnP system are introduced in this section. Next, we study the safety of six clinical scenarios including airway laser surgery, laparoscopic cholecystectomy and four clinical scenarios provided by the MD PnP program (Section 6.4). The examples are used to illustrate the four steps in our safety analysis procedure. Finally, after studying the examples, we discuss a generic version of our safety analysis approach in Section 6.5.

6.1 The MD PnP System Overview

In this section, we first look at the overall architecture of the MD PnP medical system and the associated safety requirements.

6.1.1 Overall System Architecture

As shown in Figure 6.1, the MD PnP system consists of four parts: 1) medical device 2) MD PnP device adapter, which allows the medical device to interface with the MD PnP system 3) MD PnP

[1]The definition of open-loop safety is presented in Section 6.3.

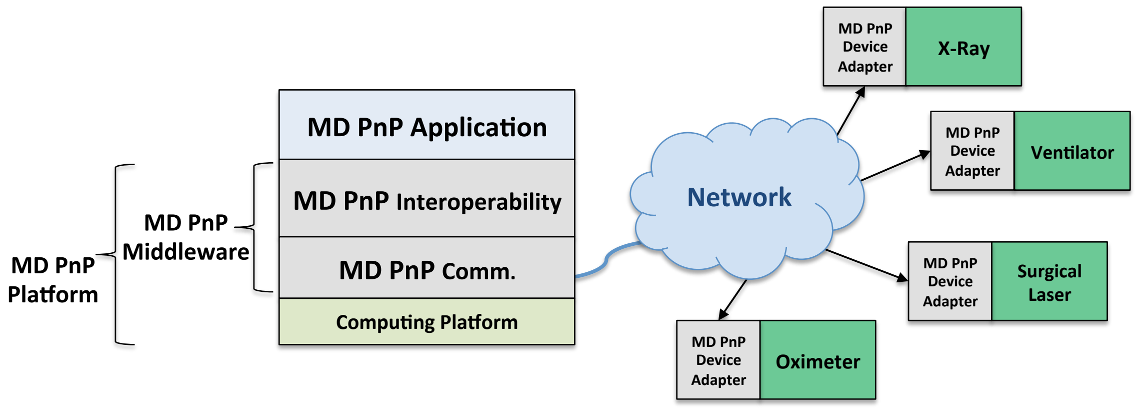


Figure 6.1: MD PnP System Overview

clinical application and 4) the MD PnP middleware that supports the communication and interoperability of medical devices [2]. In this report, we refer to the combination of computing platform (OS and microprocessor) and the MD PnP middleware as the *MD PnP platform*.

Note that the medical devices do not communicate directly with each other. In an ideal world, there would be well accepted standards on syntax and semantic for device communication. Currently legacy devices from different manufacture may provide different communication paradigm. The MD PnP middleware provides a uniform format to clinical application designers for interacting with medical devices. We assume all the MD PnP devices use the same interface description language (IDL) [84]. The middleware then translates the uniform format to specific device data and metadata format. In this way, the complexity of heterogeneous devices will be hidden from the applications and encapsulated by the MD PnP middleware.

The translation and mapping of setting from medical devices to clinical applications must be *verified to be correct* for the intended clinical use. In the following, we shall assume that developers have done so correctly.

The MD PnP middleware runs on top of commercial computing and communication equipment, which are not designed for safety critical applications. Hence, we need to consider their failure modes and the safety impacts on the intended medical applications.

[2]Interoperability refers to the ability to seamlessly connect individual devices from different vendors into integrated medical systems [49].

6.1.2 Safety Requirements

MD PnP systems, using dependable commercial platforms with sufficiently long mean time to failure, can be used safely, provided that 1) the MD PnP system will only be used as an *advisory* system and its output does not directly control any safety critical medical devices 2) the outputs will only be used to activate non-safety critical devices. Non-safety critical devices refer to the devices that their operation cannot endanger patient safety, assuming the device functions according to its specification. A safety-critical device is **life-sustaining** and can directly affect patient safety. Dependable platforms with sufficiently long mean time to failure ensure high availability and improve the system usability.

However, the notion of safety criticality for a medical device can only be considered in the context of the ***intended clinical use***. Therefore a device may be considered a safety critical device in one clinical procedure but non-safety critical in another. For example, the x-ray device in [57] is an example of non-safety critical device; the inability of the device to take an x-ray image does not endanger the patient, assuming the device functions as specified. On the other hand, if the x-ray image is used in an emergency surgery to determine the exact surgical site, the x-ray device may be considered a safety-critical device. Another example is a ventilator connected to a patient under anesthesia. If a ventilator remains off for too long, brain damage may occur. Therefore, the ventilator is considered a safety-critical device in this scenario since incorrect setting of this device can endanger patient safety.

In this report we focus on the MD PnP safety, which is a system property. However, the global MD PnP safety itself depends on the ***operational*** *safety of individual devices* constituting the system. Operational safety of medical devices is the union of *local operational safety*: when the device is used alone it will not become a causal factor of a safety hazard fault tree, and *interaction safety*: the interaction of two or more medical devices will not become a causal factor of a safety hazard in the fault tree.

As an example let us consider the airway laser surgical scenario. The main devices of the MD PnP system for this scenario include an oxygen controller and a surgical laser. To ensure local safety of the oxygen controller the device must not be $puased$ for too long that it causes brain hypoxia. However, to ensure the safe interaction of the two devices in the context of airway laser surgery, where the laser is applied to the oxygen pathway, the oxygen controller must be $puased$ when the laser is $activated$

to prevent fire. Interaction safety may create additional challenges, which must be addressed when analyzing the safety of MD PnP systems.

In the next section, we discuss the requirements for the four main elements of the MD PnP systems, when the system output is used to *directly* control safety critical devices.

Medical Device Requirements

There are two categories of safety requirements regarding medical devices. One category focuses on design safety and the other involves operational safety.

Medical Device Design Safety: The concept of safety in medical device design has been broadened from the simple, ***basic safety*** considerations to include ***essential performance*** matters [85]. As noted by the recent editions of *IEC 60601-1*[3], safety standard for medical electrical equipment, many of medical devices are life-sustaining or critical for correct diagnosis. A device which does not perform properly can present an unacceptable risk, and hence endanger patient safety. For example, a defibrillator that will not discharge may be considered unsafe. Similarly, incorrect diagnosis resulting from a monitoring device malfunction can lead to unsafe or ineffective treatment.

According to this standard, ”basic safety provides protection against *direct physical hazards* when medical electrical equipment is used under normal or other reasonably foreseeable conditions, (e.g. mechanical strength, leakage current and fire safety)” while, ”essential performance requirements limit or establish critical parameters such as the frequency response of a diagnostic electrocardiograph and the output energy of a cardiac defibrillator”, which presents an unreasonable risk should this performance fail.

Within the scope of this study we assume medical devices function as specified and satisfy their *safety* and *essential performance* requirements. To ensure safety of the MD PnP system when medical devices are malfunctioning is a topic to be addressed in the future.

Medical Device Operational Safety: In addition to design safety, there is another aspect to medical device safety, which is how to ensure *operational safety* of medical devices during the patient care process. The operational safety constraints of medical devices are context dependent and may change

[3]The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees. IEC 60601-1 is the parent safety standard for medical electrical equipment.

dynamically during the patient care process. The context refers to the information on the patient state, status of all medical devices in the system, clinical procedure, etc. As explained in Section 6.1.2, operational safety of medical devices is the union of *local* operational safety and device *interaction safety*. In this report, we focus on operational safety of medical devices when analyzing the global safety of the MD PnP systems.

Although we assume that the medical device design is safe and it satisfies the basic safety and essential performance requirements, for a device to be operating as part of the MD PnP system, it must also meet the following requirements:

- The device must provide the capability to be controlled and monitored.
- The device must be equipped with local display that shows the current device operational status such as ventilation mode, infusion rate, medication dosage, etc. The device must also display what component is controlling the device (ex: MD PnP through remote configuration panel, or human operator through local control buttons). In other words, the locus of control must be clear. The local display is specifically critical for manual execution of fail-safe steps for each device. In addition, explicit notice for overwrite by human operator must be put on the device local display.

MD PnP Device Adapter Requirements

As described in the ICE standard, a legacy medical device must be equipped with an adapter to allow the device to interface with the MD PnP middleware, so that the device can be under supervisory control. The existing medical/ health device communication standards, such as *ISO/IEEE 11073* [86], [87], although necessary for designing medical device interfaces, are not sufficient when developing ICE applications to control and coordinate devices.

We assume a crash failure semantic for this component, i.e. it can only fail by crashing. The correctness of MD PnP device adapter must be verified to ensure that control or monitoring values communicating through the adapter would not be altered. We discuss the case, in which the adapter crashes, in later sections.

Since medical device adapters play a critical role in safe use of MD PnP systems, we are required to discuss the ***dependability or reliability*** of this component, which describes its ability to function as specified under stated conditions for a specified period of time. The device adapter reliability affects the *open-loop safe* solution that can be used for the MD PnP system. We discuss the details of open-loop safety in Section 6.3. We consider two classes of reliability for the device adapters:

Highly Reliable: the device adapter must satisfy an appropriate meantime to failure (MTTF) requirement to ensure safety of an intended medical application:

- $MTTF \geq N_{Highly\ Reliable} \times Operation\ Window$

where $Operation\ Window$ denotes the maximum operation time for the intended clinical use and $N_{Highly\ Reliable}$ represents the dependability requirement, which can also be adjusted for each clinical procedure. For example, under the commonly used exponential reliability model[4], $N_{Highly\ Reliable} = 1,000,000$ implies one failure per million operations on average. This number should be sufficiently high for the intended medical procedure, so that the benefit significantly outweighs the risk. Our design for qualified safety critical components assumes that their probability of failure during the window of operation is virtually zero.

Reliable: the device adapter must satisfy the following meantime to failure (MTTF) requirement:

- $N_{Reliablel} \times Operation\ Window \leq MTTF < N_{Highly\ Reliable} \times Operation\ Window$

where a lower number, perhaps, $N_{Reliable} = 10,000$, is acceptable. In medical domain, the system reliability requirements depend on the intended clinical use. This is unlike the avionic domain, in which fixed reliability values are often recommended by FAA.

MD PnP Platform Requirements

We call the combination of a computational platform and the MD PnP middleware the *MD PnP platform*. We consider the following requirements:

[4] $R(t) \sim Exp(\frac{-1}{MTTF/Operation\ Window})$

Data Transformation Correctness: MD PnP middleware logic must be verified. For example, the syntax and semantic translation and mapping of settings of devices from different vendors must be *verified to be clinically correct* before usage. In the following, we shall assume that designers have done so correctly.

Data Integrity: This requirement ensures that the data and supervisory commands going through the MD PnP platform, before being delivered to medical devices, would not be corrupted in a way that cannot be detected.

It is the MD PnP system developer's responsibility to validate these requirements. However, commercial computational platforms and networks are not designed for safety critical applications. Hence, we assume that they may crash at any instant. We will use the **crash failure semantic** for the MD PnP platform in the safety analysis in the rest of this report.

It is quite common that commercial platform vendors issue software or firmware upgrades; designers should not install upgrades without re-verifying the failure mode assumptions. For example, upgrades can create backward compatibility issues and unintended bugs not considered in the initially assumed failure modes for platform or MD PnP software. In cases the upgrades are applied and the assumptions on failure semantic do not hold anymore, the safety analysis must be re-examined.

MD PnP Application Requirements

When the MD PnP system in used to directly control safety critical medical devices, it will be the developer's responsibility to ensure that the failure of the MD PnP system cannot be a causal factor of safety hazards. We consider the following safety requirements:

Computational Correctness: For each MD PnP clinical application, the *specification* must be validated and the *implementation* correctness must be verified. For example, if the application implements a safety interlock or computes closed loop control values, the specification of the intended capabilities, fundamental use and desired outputs must be validated and the correctness of safety interlock or control logic implementation must be verified.

Data Integrity: This requirement ensures that the MD PnP application does not perform unspecified transformation or change measurements and status information received from medical devices in a way that cannot be detected.

Symbol	Description
MD PnP *Supervisor* is Unable to Coordinate Devices	State Representation
MD PnP Platform Crashes	Event Representation

Table 6.1: Common Symbols in Extended Fault Tree

In rest of this report we will assume that developers have correctly verified these requirements. However, the MD PnP platform and/or network may crash and cause the MD PnP application to crash as well. Assuming that the computational correctness and data integrity requirements have been verified by the developers, we only consider **crash failure semantic** for this MD PnP component.

Finally, an error-free safety analysis is seldom an easy task. The MD PnP program cannot be responsible for the correctness of safety analysis performed by designers. In the next section, we present the general safety analysis procedure followed in this report. Before explaining the different steps of safety analysis, however, we present a brief introduction on extended fault tree analysis.

6.2 Introduction to Fault Tree Analysis

To make this report self-contained, we review the main components of a fault tree in this section. Fault tree analysis is a commonly used method to ensure no undesired event in the system can compromise the system-wide safety. We also use Fault Tree analysis to demonstrate the behavior of the system, which leads to the hazardous and unsafe situations. Safety cannot be modeled by fault trees, which are simply representations of causal chains. Therefore, we use extended fault tree (EFT), which is the combination of classic fault trees and state/event semantics [88].

6.2.1 Extended Fault Tree Analysis

The main goal in extended fault tree analysis is adding the capability of expressing state dependencies or temporal order of events. This is necessary to describe a system's behavior and analyze its reliability and safety-related issues. In the extended fault tree, there are two type of nodes, state node and event node. Table 6.1 shows the common symbols of an event and a node of the fault tree representation. *States* describe conditions that last over a period of time whereas *events* are sudden

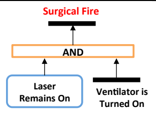
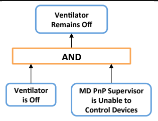
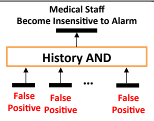
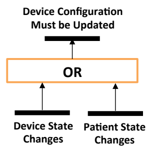
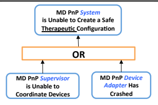
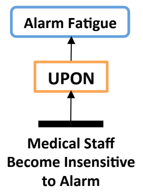
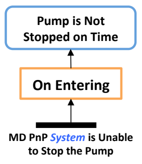
Gate Name	Symbol	Description
AND	 Surgical Fire AND Laser Remains On Ventilator is Turned On	If a certain event happens *while* the system is in a given state, then immediately after the output event happens.
AND	 Ventilator Remains Off AND Ventilator is Off MD PnP Supervisor is Unable to Control Devices	This gate has generalization semantic.
History AND	 Medical Staff Become Insensitive to Alarm History AND False Positive False Positive ... False Positive	The output event occurs immediately after the last of its input events has occurred, provided that all other input events have occurred before. The input events can occur at any order.
OR	 Device Configuration Must be Updated OR Device State Changes Patient State Changes	Occurrence of any of the input events causes the output event to occur.
OR	 MD PnP System is Unable to Create a Safe Therapeutic Configuration OR MD PnP Supervisor is Unable to Coordinate Devices MD PnP Device Adapter Has Crashed	This gate has generalization semantic: The output state can be regarded as a collective alias for the input states.
UPON	 Alarm Fatigue UPON Medical Staff Become Insensitive to Alarm	*Upon* is a converting gate; it models the passage from an event to a state, which has the meaning of referring to the state after the event has happened for the first time. The input is an event and output is a state.
On Entering	 Pump is Not Stopped on Time On Entering MD PnP System is Unable to Stop the Pump	*On Entering* is a converting gate; it models the passage from a state to an event, which has the meaning of referring to the point in time where the state is entered. The input is a state and output is an event.

Table 6.2: Gates Commonly Used in Extended Fault Tree

phenomena, including state transitions. In EFT, round rectangles represent states and solid bars denote events. Directed edges with bold arrowheads mark the causal edges. Table 6.2 depicts the gates and notations used in fault tree analysis in this report. More details about extended fault tree can be found in [45], [44], and [88].

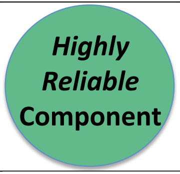
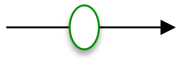
Symbol	Semantic
	Component Satisfies *Safety-Critical* Requirements
	Non-Causal Directed Edge

Table 6.3: Symbols for Newly-Introduced Extended Fault Tree Components

6.2.2 Context-Aware Extended Fault Tree Analysis

We introduce two additional elements to extended fault tree to provide more contextual information. The first one is the ***Highly Reliable Component***. This elements represent a component in the system that meets the reliability requirement as described in Section 6.1.2: its meantime to failure must be much longer than the operational window so that the probability of failure is deemed to be sufficiently low for the intended clinical use. The next element is a ***Non-Causal Directed Edge***, which represent the cases in which system component may crash but its failure is not a causal factor of safety hazard. Including such components in fault tree analysis provides additional information on system behavior. Without the new elements, absence of faulty states or events of a component in tree can mean either it does not contribute to safety hazard, or the designer has missed considering the component in safety analysis. Including *highly reliable components* and/or *non-causal edges* in the tree can provide more details on the system design and the approaches taken by the designer to trim the tree or break the paths to tree root. Introducing these elements also helps with evaluating the comprehensiveness of fault tree. Table 6.3 presents the symbols used for the new components and their semantic.

6.3 Safety Analysis of the MD PnP Systems

In this document, we illustrate how to determine whether the planned MD PnP system usage is a causal factor in the known safety hazards of a medical scenario. Should the MD PnP usage be a causal factor of safety hazards, we provide guidelines to transform the configuration so that it is no longer a causal factor. To ensure safe use of the MD PnP system, we follow a 4-step safety analysis procedure:

- **S1**: Enumerate the safety hazards of the medical scenario and draw the extended fault trees for

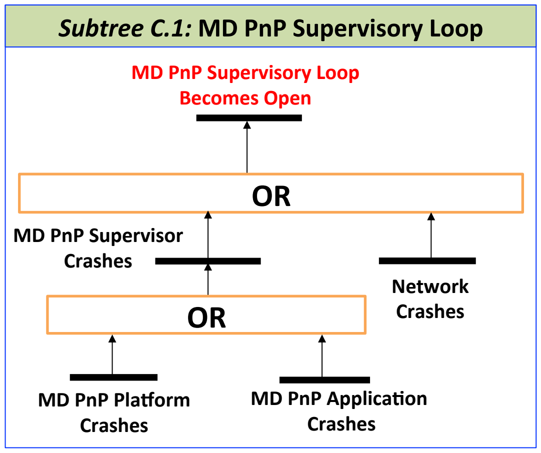

Subtree C.1: MD PnP Supervisory Loop
MD PnP Supervisory Loop Becomes Open
OR
MD PnP Supervisor Crashes
Network Crashes
OR
MD PnP Platform Crashes
MD PnP Application Crashes

Figure 6.2: Fault Sub-Tree for MD PnP Supervisory Loop Crash

each hazard.

- **S2**: Identify whether any MD PnP component is a causal factor of the hazards.
- **S3**: Recommend a fail-safe solution if any MD PnP component is a causal factor of those hazards. (without a fail-safe solution users may choose not to use the MD PnP system if the risks are not outweighed by the benefits)
- **S4**: Redraw the extended fault tree for any possible fail-safe solution to ensure that the MD PnP platform/network failure path to the root is removed.

The first and fourth step involve drawing fault trees for enumerated safety hazards of each clinical scenario. In some cases, the trees can get complex, which makes them less readable. To prevent this problem, we take a modular approach, in which we introduce sub-trees that can be re-used in several fault tree analyses. When drawing the trees for different clinical scenarios we only show the root of subtrees used and their number in a green bubble for the readers to refer back for more details. The first subtree we present is shown in Figure 6.2 which represents the series of events leading to the MD PnP supervisory loop crash. In the third step of safety analysis, we provide guidelines and open-loop safe solutions to transform the system configuration so that no MD PnP component is a causal factor.

Open Loop Safety: During normal operation, medical devices are monitored and coordinated by the supervisor. If either the supervisor or the communication network fails, the supervisory control loop is

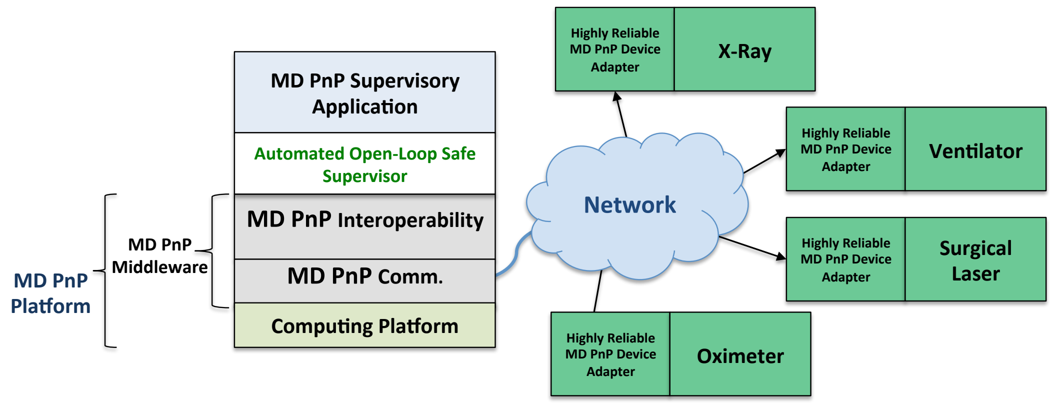

MD PnP Supervisory Application
Automated Open-Loop Safe Supervisor
MD PnP Interoperability
MD PnP Comm.
Computing Platform
MD PnP Middleware
MD PnP Platform
Network
Highly Reliable MD PnP Device Adapter
X-Ray
Highly Reliable MD PnP Device Adapter
Ventilator
Highly Reliable MD PnP Device Adapter
Surgical Laser
Highly Reliable MD PnP Device Adapter
Oximeter

Figure 6.3: Automated Open-Loop Safe MD PnP System Overview

said to be open. By open loop safety, we mean that even if the supervisory loop opens, the MD PnP system still cannot be a causal factor of a safety hazard. Note that there are long-term and near-term open-loop safe solutions:

- The long-term solution requires device makers to support simple *timed safety actions*. Otherwise, the MD PnP device adapter must provide such capability. In this case, the device adapter must be ***highly reliable***, for the open-loop safety solution to work. We call this approach **Automated Open-Loop Safety** (see Section 6.3.1).

- The near-term solution for existing medical devices is to use a component which includes a recording device and a watchdog timer to remind users to perform the *timed safety actions*. In this case the medical device adapters must only be ***reliable***; however this added component must be ***highly reliable***. The device adapter cannot be trusted to move the medical device into a safe state when the supervisory loop fails. Hence, open loop safety protocols are designed to reliably pass the control back to the medical staff. We call this approach **Integrated Human Machine (IHM) Open-Loop Safety** (see Section 6.3.2).

6.3.1 Automated Open-Loop Safety

As shown in Figure 6.3, the automated open-loop safe MD PnP system consists of the following components:

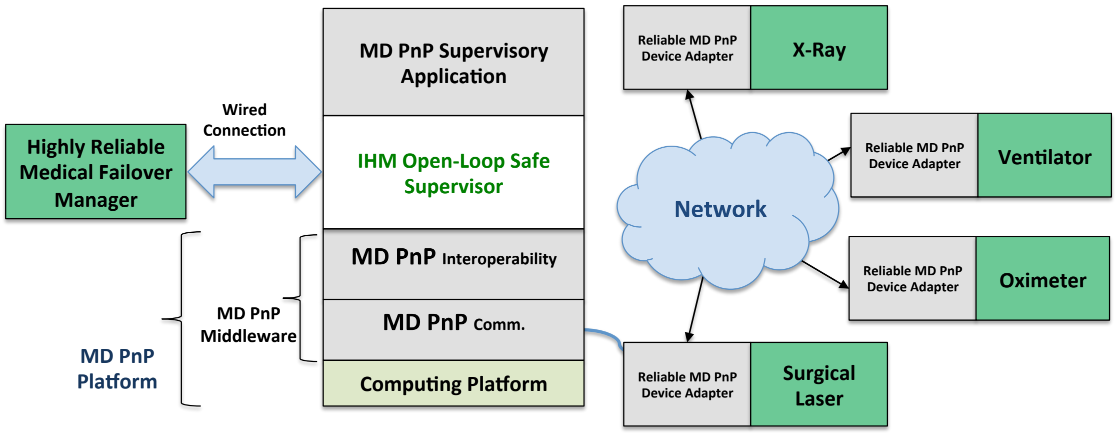


Figure 6.4: IHM Open-Loop Safe MD PnP System Overview

Automated Open-loop safe supervisor: This supervisory logic is added to the application layer and runs on top of the MD PnP platform. The commands received from the MD PnP supervisor go through the open-loop safe supervisor before being sent to the medical devices. The commands are decomposed into a sequence of timed actions, which ensure that the devices move to a safe state by taking these actions when the supervisory loop is open. At any point during the communication of these messages, the supervisory loop can become open due to network failure or crash of the MD PnP platform. Note that this will not causes any safety hazards since no commands will be sent unless the corresponding timed safety command has been sent successfully. This protocol is ***any-step safe***.

Highly reliable medical device adapters: This component ensures the timed safety commands[5] are executed by the device when the MD PnP supervisory loop is open. It is assumed that the MD PnP device adapter is *highly reliable*; otherwise, automated open-loop safety cannot be guaranteed.

6.3.2 IHM Open-Loop Safety

As shown in Figure 6.4, the automated open-loop safe MD PnP system consists of the following components:

Medical Failover Manager (MFM): This component must be ***highly reliable***. It shall be a simple and verifiably reliable storage and display device that stores and displays the commands sent by the MD

[5]A timed safety command is a command to execute a safety action at the expiration of a timer set by the application, e.g., resume ventilation in 30 seconds. An example is in Page 108.

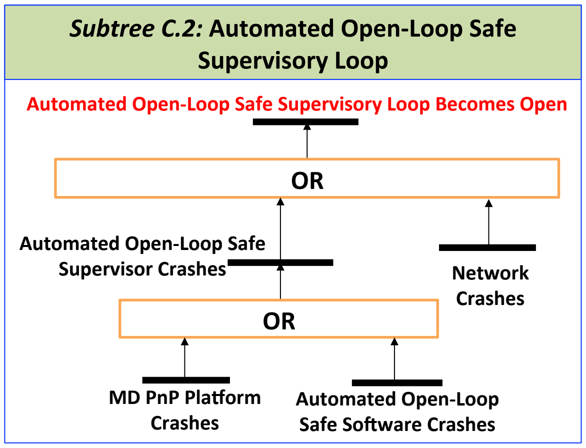
PnP supervisor. Similar to avionic systems, the capability of having a record of what the system has been doing will have great safety and legal values. In addition, the MFM has a configurable watchdog timer that will issue an alarm, should the MD PnP supervisor stop sending heartbeats to the MFM. Once the alarm is sounded, medical staff can check the MFM and know what the most recent command is, which may or may not be executed, and the recent history. This allows medical staff to take safety actions in an informed way. The MFM component consists of **1)** a highly reliable storage and display device, **2)** a watch dog timer and alarm, **3)** an interface with the MD PnP system. If the supervisor fails to deliver heart beats to the timer, an alarm will sound indicating that the MD PnP supervisory loop is open.

Integrated Human Machine (IHM) open-loop safety supervisor: In addition to the *highly reliable* ***MFM***, the near term solution includes a supervisory component, which is responsible for executing the open-loop safety protocol steps. In this approach, we use timed alarms to ask medical staff to move the system into a safe state, since we assume device adapters are **not *highly reliable*** and therefore cannot be trusted to move the medical devices to safe states. The protocol includes the following steps:

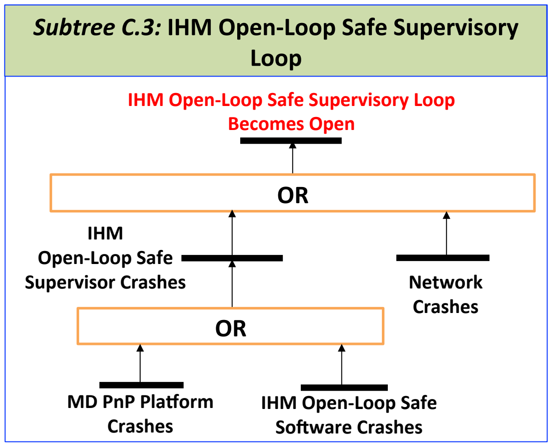
- At each step, the IHM open-loop safety supervisor first sends the command (received from the MD PnP supervisor) to the MFM and waits for an acknowledgment before sending the command to the medical device. This way, when the MD PnP system crashes and the alarm sounds by MFM, medical staff know the command history and the most recent command that the supervisor attempted to execute, greatly facilitating the medical staff's take over task. For commands that will put the device into a transient safe state, the command should be annotated with the open loop safe action that a staff should perform manually. For example, the "pause" command sent to the MFM is annotated as $pause\ at\ time\ t,\ (to\ be\ followed\ by\ "resume"\ command\ at\ t+30sec)$.

- After receiving an acknowledgment from the MFM, the IHM open-loop safety supervisor sends the command to the medical device and waits for an acknowledge from the device. If an acknowledgment is not received in time, an alarm will sound to let medical staff know that the device is not responding.

Figure 6.5(a) shows the sub-tree for automated open-loop supervisory loop crash and Figure 6.5(b) shows the sub-tree for IHM open-loop supervisory loop crash.


Subtree C.2: Automated Open-Loop Safe Supervisory Loop
Automated Open-Loop Safe Supervisory Loop Becomes Open
OR
Automated Open-Loop Safe Supervisor Crashes
Network Crashes
OR
MD PnP Platform Crashes
Automated Open-Loop Safe Software Crashes

(a) Automated Open-Loop Safe


Subtree C.3: IHM Open-Loop Safe Supervisory Loop
IHM Open-Loop Safe Supervisory Loop Becomes Open
OR
IHM Open-Loop Safe Supervisor Crashes
Network Crashes
OR
MD PnP Platform Crashes
IHM Open-Loop Safe Software Crashes

(b) IHM Open-Loop Safety

Figure 6.5: Subtrees for Supervisory Loop Crash

We recommend the record keeping in general. An accurate record of what the MD PnP supervisor has done will be invaluable for post operation analysis, especially settling disputes, if any. To this end, MFM's record system should be protected from tampering by unauthorized users.

In the next section, we illustrate how to use each open-loop safety approach to ensure that no MD PnP component is a causal factor of known safety hazards in the context of several clinical scenarios.

6.4 MD PnP Clinical Scenarios

In this section, we study six clinical scenarios and demonstrate the process of safety analysis of the MD PnP system using extended fault trees in each case. For more details on these scenarios refer to [89], [90] and [91]. We assume for each scenario, the clinical application specifies what kind of response it expects form each medical device. In other words, the device composition must be definitive for each application.

6.4.1 Laparoscopic Cholecystectomy

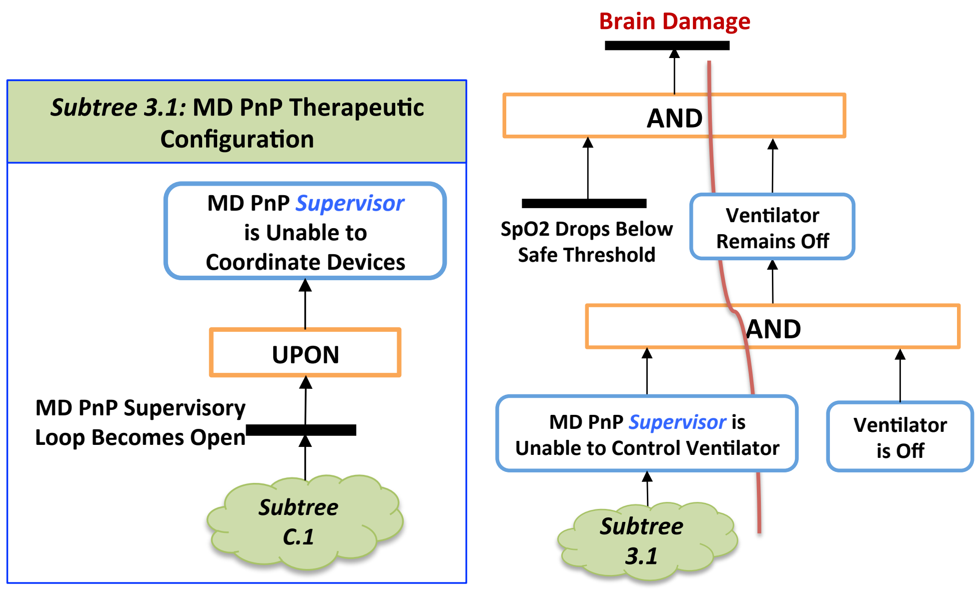
A 32-year-old woman had a laparoscopic cholecystectomy (gall bladder removal) performed under general anesthesia. At the surgeon's request, a plain film x-ray was shot during a cholangiogram [bile duct x-ray]. The anesthesiologist stopped the ventilator for the film. The x-ray technician was unable to remove the film because of its position beneath the table. The anesthesiologist attempted to help her, but found it difficult because the gears on the table had jammed. Finally, the x-ray was removed, and the surgical procedure resumed. At some point, the anesthesiologist glanced at the EKG and noticed severe bradycardia. He realized he had never restarted the ventilator. The ventilator is typically stopped for 20 to 60 seconds to prevent motion-induced blurring of the image. This patient ultimately expired. [92].

There are two approaches to prevent the motion-induced blur in the x-ray. If we use the MD PnP system to pause the ventilator, then the MD PnP system becomes safety critical and we will illustrate how to ensure open-loop safety. An alternative, in which the MD PnP system does not need to be safety critical [57], is suggested.

The supervisor uses information from the ventilator to decide when to trigger the x-ray. The synchronization algorithm defines exactly how this decision is made according to respiratory cycle as a function of pressure over time. The pressure increases until the end of inspiration (at time T after start of breath), at which point it drops off quickly through expiration. There is usually a pause between the end of exhalation and the start of the next breath. For this case study, we want to support taking an x-ray when the lung was not moving significantly. This occurs when the patient is relatively still at the peak of inspiration or between the end of expiration and the start of the next breath. An exposure is possible if the time the patient is still exceeds the time needed for the exposure plus the latency between triggering the x-ray and the actual exposure.

Under this approach, the MD PnP system is not safety critical because it does not send stop commands to the ventilator, which is a safety-critical device. We recommend this approach whenever it is applicable. However, should pausing the ventilator be needed and users want to use the the MD PnP system to manage the pause directly, then the MD PnP system becomes safety critical. Since the MD PnP supervisor sends commands directly to the safety critical ventilator, this case shall be subject to the 4-step safety analysis process. The first step is to determine the safety hazard(s) in this scenario and perform the fault tree analysis (**S1**).

Following the modular approach, we first present the subtree for the MD PnP therapeutic configuration, $subtree$ 3.1, in Figure 6.6(a). Next, we use this subtree to represent the series of fault events and states regarding the main elements of the MD PnP system, which can contribute to brain damage


Subtree 3.1: MD PnP Therapeutic Configuration
MD PnP Supervisor is Unable to Coordinate Devices
UPON
MD PnP Supervisory Loop Becomes Open
Subtree C.1
Brain Damage
AND
SpO2 Drops Below Safe Threshold
Ventilator Remains Off
AND
MD PnP Supervisor is Unable to Control Ventilator
Ventilator is Off
Subtree 3.1

(a) MD PnP Therapeutic Configuration Subtree (b) Extended Fault Tree for Brain Damage

Figure 6.6: Extended Fault Tree Analysis of Brain Damage in the MD PnP System

in Figure 6.6(b).

In theory, the supervisor can send an atomic command $pause\ and\ then\ resume\ at\ time\ t$, which will be safe if the approved ventilator can execute atomic transactions. This is the preferred method. However, currently ventilators can only execute simple commands such as turn on and turn off. Hence, users may use pause command followed by a resume command later at time t. In this case, crash of the MD PnP platform or the network opens up a path to the root of tree and leads to brain damage. Moreover, any error in the MD PnP supervisory logic, which is responsible for coordinating the x-ray and ventilator devices can lead to an unsafe ventilator configuration; this should be mitigated. The dotted red lines on the extended fault tree indicates these paths (step **S2**).

To break such paths to the root and ensure system safety despite the failure of these components, an ***open-loop safe*** approach is needed (step **S3**). Next, we present the automated and IHM open-loop safety solutions for this clinical scenario.

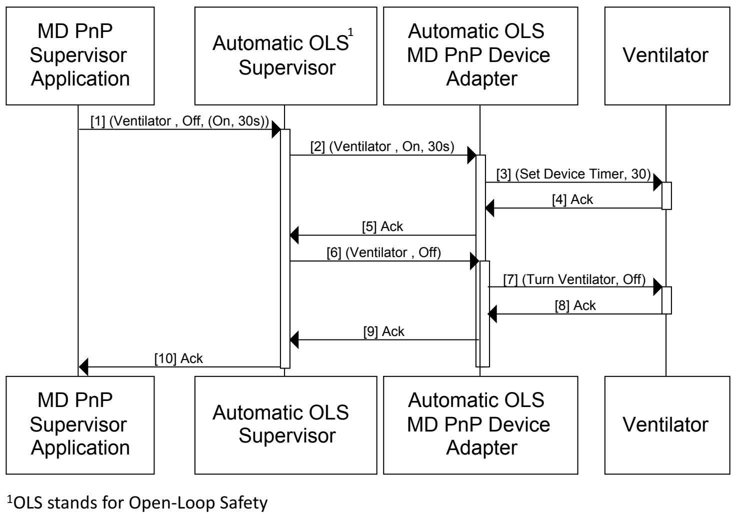
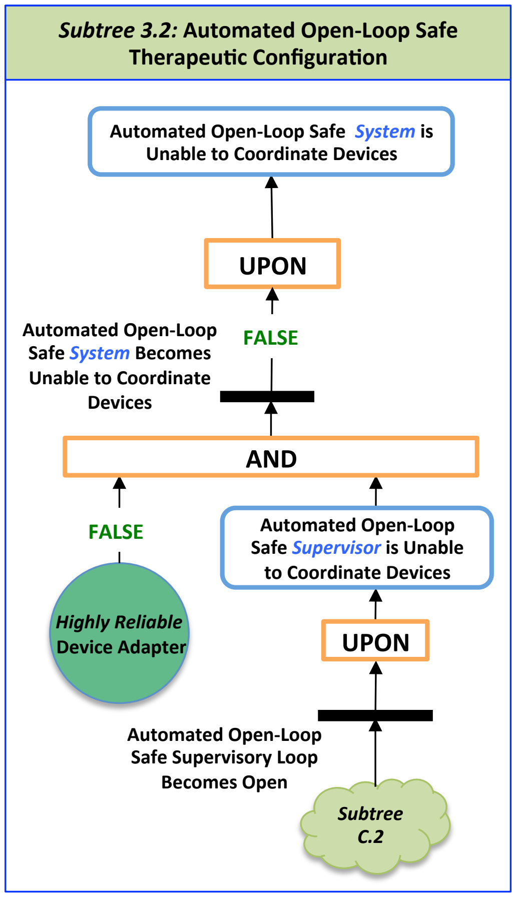

MD PnP Supervisor Application
Automatic OLS[1] Supervisor
Automatic OLS MD PnP Device Adapter
Ventilator
[1] (Ventilator , Off, (On, 30s))
[2] (Ventilator , On, 30s)
[3] (Set Device Timer, 30)
[4] Ack
[5] Ack
[6] (Ventilator , Off)
[7] (Turn Ventilator, Off)
[8] Ack
[9] Ack
[10] Ack
MD PnP Supervisor Application
Automatic OLS Supervisor
Automatic OLS MD PnP Device Adapter
Ventilator
1OLS stands for Open-Loop Safety

Figure 6.7: Sequence Diagram for Automated Open-Loop Safe for Safe Ventilator Configuration

Automated Open-loop Safe Coordination of X-ray and Ventilator

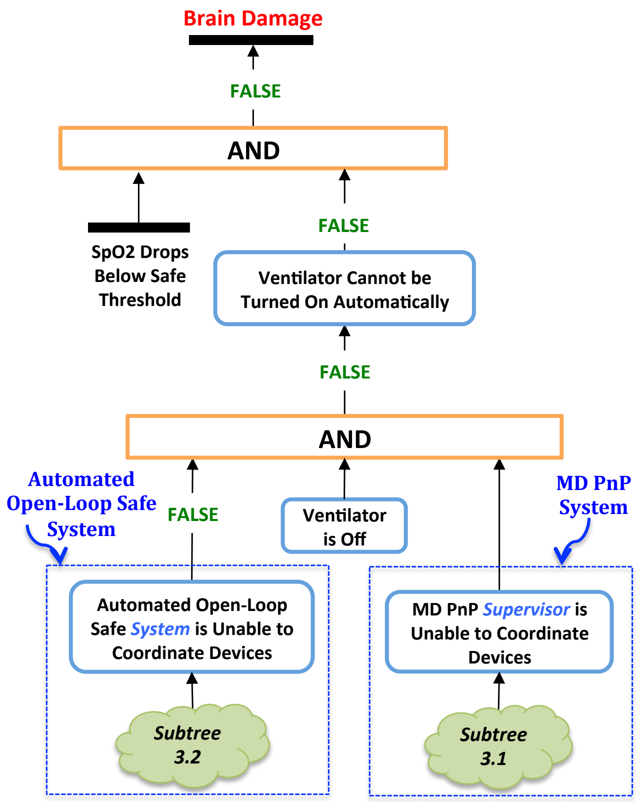
In this section we present the automated open-loop safe approach to prevent crash of the MD PnP supervisory loop to cause brain damage in the laparoscopic cholecystectomy scenario (step **S3**). When the MD PnP supervisor intends to send a command to turn off the ventilator for a period of time, a sequence of timed actions are sequentially sent by the open-loop safe supervisor. In this case, a command to resume the ventilator at time t is sent *before* sending the command to turn off the ventilator. The open loop safe protocol ensures that the timed safety command is received by the device before sending the ”turn off” command. The ”turn off” command puts the ventilator in a transient safe state, because if the ventilator remains off for too long, patient’s blood oxygen level can drop below the safe threshold and brain damage may happen.

The commands for timed actions are sent over the network and received by the open-loop safety device adapter, since existing ventilators do not support timer based actions. When the ”resume ventilator at time t” command is sent to the MD PnP device adapter, if the ventilator has timer capability then the device adapter, which must be *highly reliable*, sets the device timer; otherwise, a timer is created and set by the adapter, which triggers the automatic resumption of ventilator upon expiration. After setting the timer (through either of the two approaches), an acknowledgment is sent to the open-loop



Subtree 3.2: Automated Open-Loop Safe Therapeutic Configuration
Automated Open-Loop Safe System is Unable to Coordinate Devices
UPON
FALSE
Automated Open-Loop Safe System Becomes Unable to Coordinate Devices
AND
FALSE
Automated Open-Loop Safe Supervisor is Unable to Coordinate Devices
Highly Reliable Device Adapter
UPON
Automated Open-Loop Safe Supervisory Loop Becomes Open
Subtree C.2

(a) Automated Open-Loop Safe Therapeutic Configuration Subtree



Brain Damage
FALSE
AND
FALSE
SpO2 Drops Below Safe Threshold
Ventilator Cannot be Turned On Automatically
FALSE
AND
Automated Open-Loop Safe System
FALSE
Ventilator is Off
MD PnP System
Automated Open-Loop Safe System is Unable to Coordinate Devices
MD PnP Supervisor is Unable to Coordinate Devices
Subtree 3.2
Subtree 3.1

(b) Extended Fault Tree for Brain Damage

Figure 6.8: Brain Damage Extended Fault Tree for Automated Open-Loop Safe MD PnP System

safe supervisor. At this point, when it is assured that a timer to resume the ventilator is set, the command to turn off the ventilation is sent. Figure 6.7 shows the sequence of messages exchanged among the different components of the open-loop safe MD PnP system. The timer value in this example is only a nominal value; in general, the timer value for different actions can be updated as more recent information on patient state and status of medical devices become available.

At any point during the communication of these messages, the supervisory loop can become open due to network failure or crash of the MD PnP platform. This will not causes a hazard since the command to turn off the ventilator will not be sent unless the resume timer has been set successfully. This protocol is *any-step safe*. If the ventilator device does not have a timer feature, the open-loop safe protocol depends on the MD PnP device adapter to set up the timer and perform the timed safety action. In this case it is assumed that the MD PnP device adapters are *highly reliable* and meet the safety-critical requirements; otherwise, open-loop safety cannot be guaranteed. We recognize that such highly reliable adapters will not be available in the near term, hence an alternative ”medical staff in the loop” safety solution is provided in the next section.

For the last step of our safety analysis procedure, we redraw the extended fault tree that can lead to brain damage. Figure 6.8(b) shows the updated fault tree, which includes the unsafe therapeutic configuration subtree for automated open-loop safe system, $Subtree\ 3.2$. This subtree, shown in Figure 6.8(a), indicates that the automated open loop safe supervisor may fail. However, the open loop safe protocol is any-step safe by definition; the open-loop safe supervisor always ensures that the MD PnP device has received the timed safety action before moving to a transient safe state. Moreover, the device adapter is *highly reliable*. Therefore, if the supervisory loop becomes open the adapter will move the device into a safe state. Assuming the initial configuration is safe, failure of supervisory loop at any step of the handshaking protocol will not result in an unsafe configuration. The safe ventilator operation is added as depicted by AND gate in Figure 6.8(b)), which results in removal of all paths to the root.

IHM Open-loop Safe Coordination of X-ray and Ventilator

Timed safety commands executed by *highly reliable* medical device adapters to enable automated open loop safety protocols is a preferred long term solution. However, highly reliable adapters may not be on the market in the near future. A simple near term solution is to include a ***MFM*** in the MD PnP system. Instead of depending on device adapters to move the system into an open-loop safe state, we only require one *highly reliable* component, the ***MFM*** to remind medical staff to move the system into an open loop safe state. With a ***highly reliable MFM***, all the medical device adapters are only required to be ***reliable***. The MD PnP supervisor should follow the IHM protocol for sending commands to the ventilator. The communication between ventilator and MD PnP supervisor in the laparoscopic cholecystectomy scenario includes the following steps:

- The MD PnP supervisor sends $pause\ and\ then\ resume\ at\ time\ t$ command to the MFM and waits for an acknowledgment; if one is not received, an alarm sounded by the supervisor.

- If an acknowledgment is received, the MD PnP supervisor then sends $pause\ ventilator$ to the device and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.

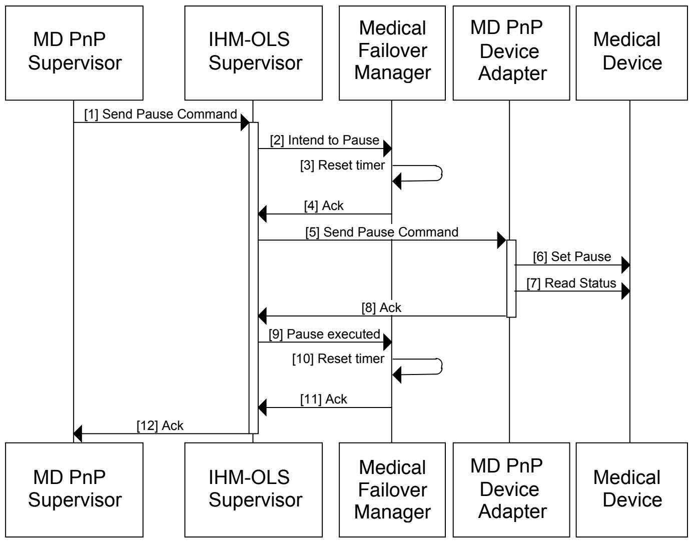
- Before pause time reaches limit, the MD PnP supervisor may send $resume\ ventilator$ command to the MFM and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.

- If an acknowledgment is received, the MD PnP supervisor sends $resume\ ventilator$ command to the device and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.

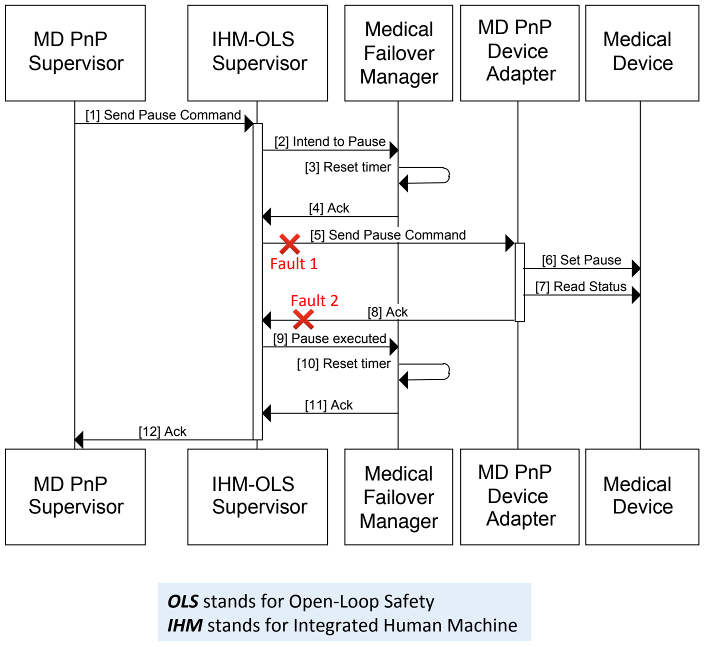
- If the MFM stops receiving heartbeats from the MD PnP supervisor, an alarm sounds to warn medical staff that the MD PnP supervisory loop has failed. The last command that supervisor attempts to execute and recent command history are displayed. To further facilitate the failure handling by medical staff, whenever the supervisor issues a command that puts the patient in a transient safe state, it should annotate the command with a timed safety action. For example, annotate the pause ventilator command with the note: pause at time $t,\ to\ be\ resumed\ by\ (t + 30sec)$

Figures 6.9(a) and 6.9(b) show the sequence of messages exchanged among the different components of the automated open-loop safe MD PnP system in the normal and crash operation modes, receptively.

At the last step of our safety analysis procedure, we redraw the extended fault tree for brain damage. First we present the necessary subtrees to re-draw the brain damage fault tree for IHM open-loop safe MD PnP system. Figure 6.10(a) shows the MD PnP therapeutic configuration subtree when only ***reliable*** device adapters are assumed, $Subtree\ 4.1$. The subtree for unsafe device configuration in the IHM open-loop safe MD PnP system, $Subtree\ 4.2$, is shown in Figure 6.10(b). Having these two, we present the brain damage fault tree for the IHM open-loop safe MD PnP system in Figure 6.11. The ***highly reliable MFM*** in $Subtree\ 4.2$ and the AND gate on top of MD PnP and IHM open-loop safe subbranches (depicted by dashed blue rectangles in Figure 6.11) results in removal of all paths to the root.



(a) Normal Mode Operation

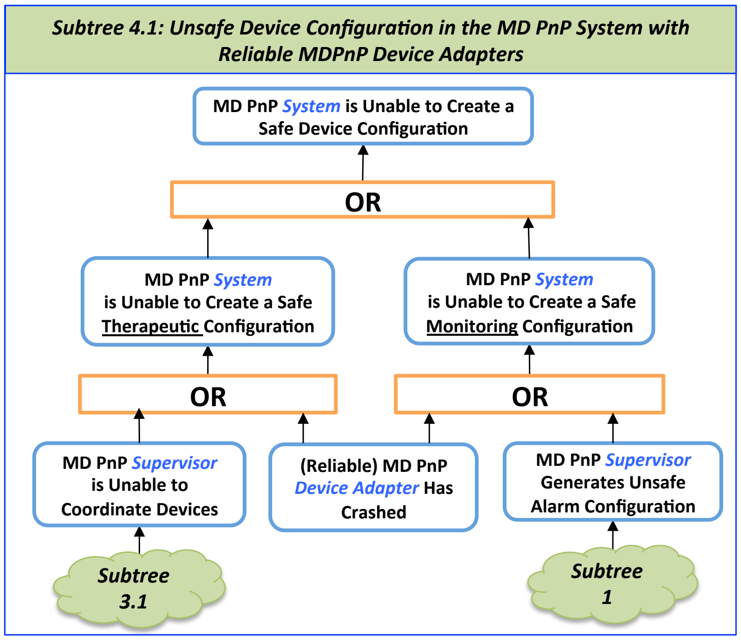


(b) Open-Loop Safe Operation

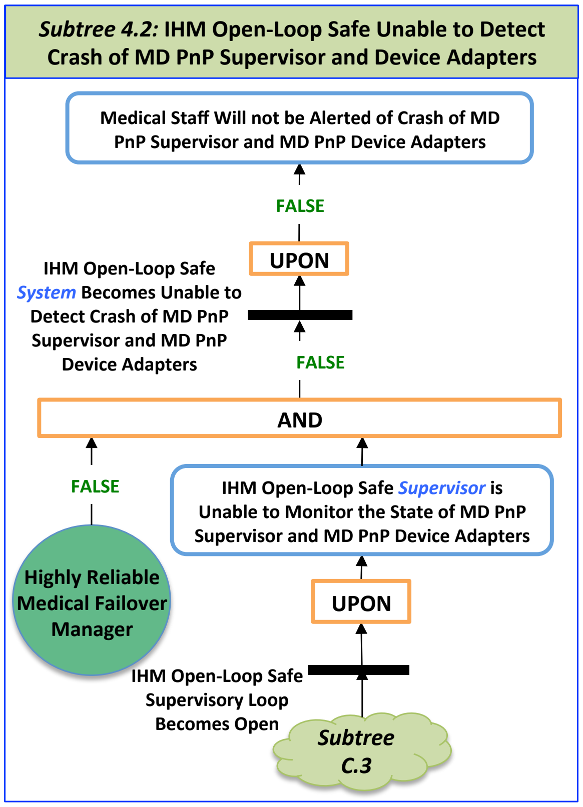
Figure 6.9: IHM Open-Loop Safe Sequence Diagram for Ventilator Configuration

6.4.2 Airway Laser Surgery

In this section we look at a slightly more complicated scenario of airway laser surgery. The airway-laser operation is performed on the patient's airway using a surgical laser when the patient is under


Subtree 4.1: Unsafe Device Configuration in the MD PnP System with Reliable MDPnP Device Adapters
MD PnP System is Unable to Create a Safe Device Configuration
OR
MD PnP System is Unable to Create a Safe Therapeutic Configuration
MD PnP System is Unable to Create a Safe Monitoring Configuration
OR
OR
MD PnP Supervisor is Unable to Coordinate Devices
(Reliable) MD PnP Device Adapter Has Crashed
MD PnP Supervisor Generates Unsafe Alarm Configuration
Subtree 3.1
Subtree 1

(a) The MD PnP Subtree


Subtree 4.2: IHM Open-Loop Safe Unable to Detect Crash of MD PnP Supervisor and Device Adapters
Medical Staff Will not be Alerted of Crash of MD PnP Supervisor and MD PnP Device Adapters
FALSE
UPON
IHM Open-Loop Safe System Becomes Unable to Detect Crash of MD PnP Supervisor and MD PnP Device Adapters
FALSE
AND
FALSE
IHM Open-Loop Safe Supervisor is Unable to Monitor the State of MD PnP Supervisor and MD PnP Device Adapters
Highly Reliable Medical Failover Manager
UPON
IHM Open-Loop Safe Supervisory Loop Becomes Open
Subtree C.3

(b) IHM Open-Loop Safe Subtree

Figure 6.10: MD PnP System with *reliable* Device Adapters and IHM Open-Loop Safe Subtrees

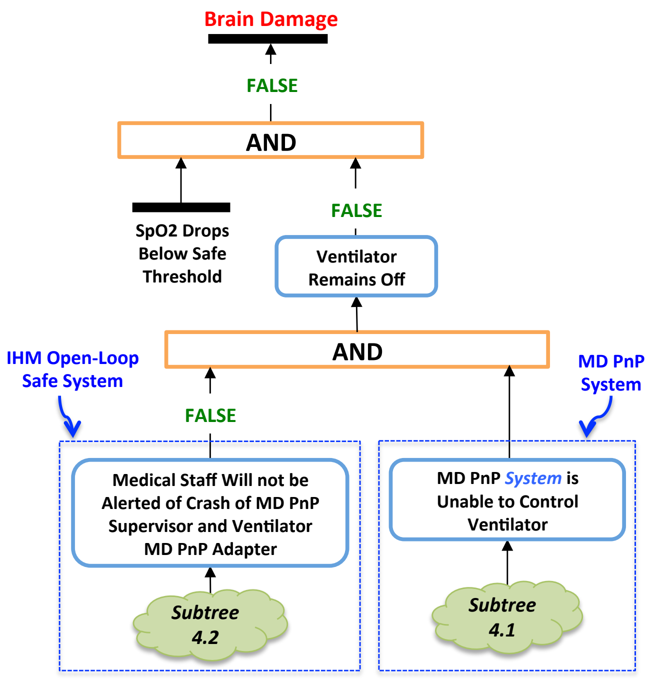

Brain Damage
FALSE
AND
SpO2 Drops Below Safe Threshold
FALSE
Ventilator Remains Off
AND
IHM Open-Loop Safe System
MD PnP System
FALSE
Medical Staff Will not be Alerted of Crash of MD PnP Supervisor and Ventilator MD PnP Adapter
MD PnP System is Unable to Control Ventilator
Subtree 4.2
Subtree 4.1

Figure 6.11: Brain Damage Extended Fault Tree for IHM Open-Loop Safe MD PnP System

anesthesia and breathing with the help of an endotracheal tube carrying air with a high concentration of oxygen. Before the surgical phase starts, 100% FiO2 supplied by a ventilator is recommended so that the surgery phase could be 1 to 5 minutes[6]. To prevent surgical fire, oxygen flow must be first blocked before the laser is in use. If the oxygen flow is blocked for too long, the patient's blood-oxygen saturation can drop below the safe threshold and brain damage can occur.

To prevent surgical fire or brain damage hazards, the MD PnP supervisor should synchronize the laser and O2 supply to ventilator. This process is performed through a series of communications between the supervisor and the medical devices. We use extended fault tree analysis to demonstrate the behavior of the MD PnP supervisory system and determine whether there is a chain of events leading to a hazardous situation. The fault tree analysis for brain damage is the same as the previous clinical scenario; therefore, we focus on surgical fire hazard. This involves interaction of two medical devices: surgical laser and ventilator. Figure 6.12 shows the extended fault tree for surgical fire. Similar to brain damage, failure of the MD PnP components or communication network can lead to surgical fire. The failure of the MD PnP supervisory loop prevents the correct commands received by the ventilator

[6]www.mc.vanderbilt.edu/documents/1anesthesiology/files/srna/Laser%20Surgery%20of%20the%20Airway.ppt

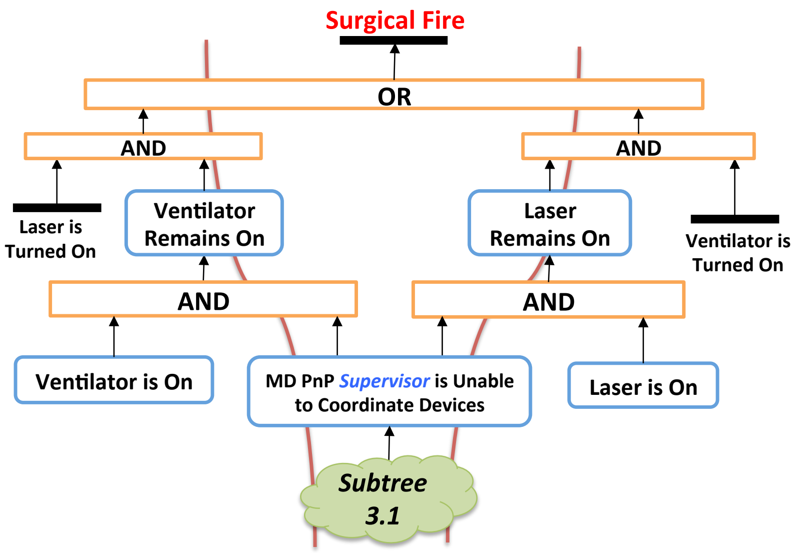


Figure 6.12: Extended Fault Tree for Surgical Fire

and surgical laser devices on time. The dotted red line shows an example of a path from an MD PnP component to the root of tree.

Automated Open-Loop Safe Approach for Airway Laser Surgical Scenario

In the airway laser surgical scenario, the MD PnP supervisor coordinates the airway laser and ventilator (oxygen supply) devices to prevent brain damage or surgical fire. However, as can be seen in the fault trees when there is a network or MD PnP supervisor failure and the supervisory loop becomes open, the patient's safety can be violated.

To prevent safety hazards when the supervisory loop becomes open, it is recommended to augment the MD PnP system with an *open-loop safety protocol*. One simple version is presented in the previous section for the laparoscopic cholecystectomy scenario. The same protocol can be used here; however, in this case there are two devices whose actions must be coordinated by the MD PnP supervisor. The open-loop safe supervisor ensures safety by sequentially sending timed action commands to the devices.

Figure 6.13 shows the sequence of communications among open-loop safety components, the MD PnP components and medical devices.

At the last step of our safety analysis procedure, we redraw the extended fault tree for surgical fire (Figure 6.14). The therapeutic configuration subtree for automated open-loop safe system, $Subtree\ 3.2$,

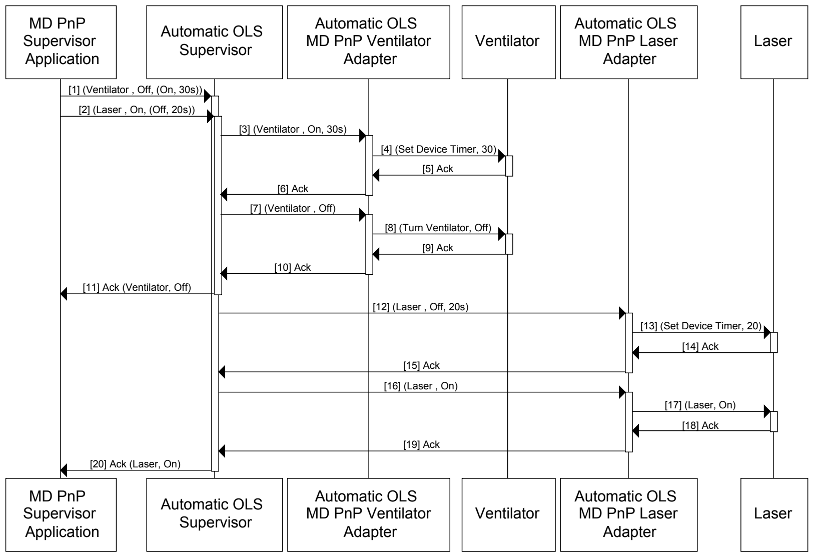

MD PnP Supervisor Application
Automatic OLS Supervisor
Automatic OLS MD PnP Ventilator Adapter
Ventilator
Automatic OLS MD PnP Laser Adapter
Laser
[1] (Ventilator , Off, (On, 30s))
[2] (Laser , On, (Off, 20s))
[3] (Ventilator , On, 30s)
[4] (Set Device Timer, 30)
[5] Ack
[6] Ack
[7] (Ventilator , Off)
[8] (Turn Ventilator, Off)
[9] Ack
[10] Ack
[11] Ack (Ventilator, Off)
[12] (Laser , Off, 20s)
[13] (Set Device Timer, 20)
[14] Ack
[15] Ack
[16] (Laser , On)
[17] (Laser, On)
[18] Ack
[19] Ack
[20] Ack (Laser, On)

Figure 6.13: Sequence Diagram of Automated Open-Loop Safe Handshaking for Airway Laser Surgery

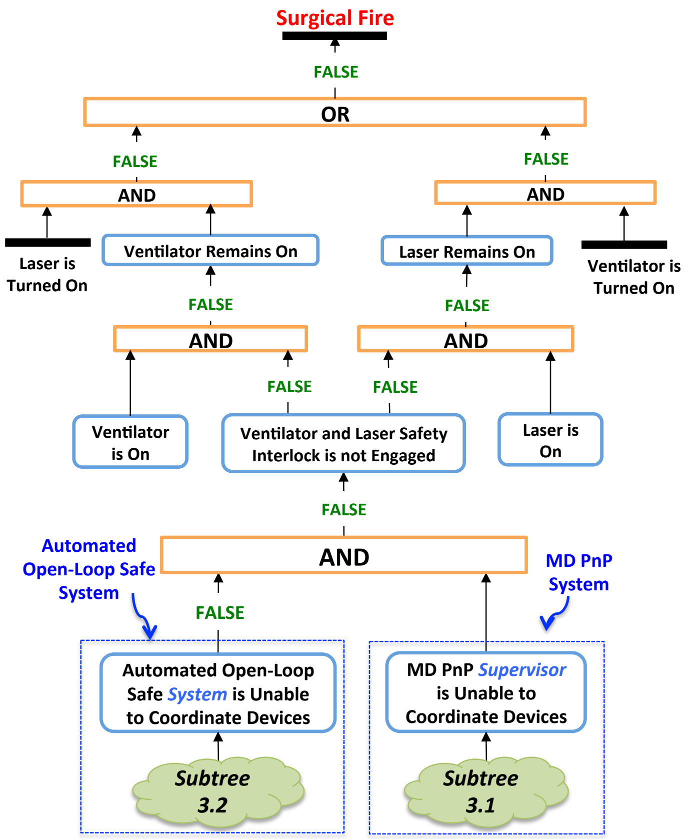


Figure 6.14: Automated Open-Loop Safe MD PnP System Fault Tree Analysis for Surgical Fire Event

is used to re-draw the fault tree. As presented on $Subtree$ 3.2 in Figure 6.8(a), the automated open loop safe supervisor may fail. However, the open loop safe protocol is any-step safe by definition; the open-loop safe supervisor always ensures that the MD PnP device has received the timed safety action before moving to a transient safe state. Moreover, the device adapters are assumed to be ***highly reliable***. Therefore, if the supervisory loop becomes open the adapter will move the device into a safe state. In conclusion, assuming the initial configuration is safe, failure of the supervisory loop at any step of the handshaking protocol will not result in an unsafe ventilator configuration. Finally, the addition of the AND gate on top of MD PnP and open-loop safe subbranches (depicted by dashed blue rectangles in Figure 6.14) results in removal of all paths to the root.

IHM Open-Loop Safe Approach for Airway Laser Surgical Scenario

As explained in the previous clinical example, timed safety commands executed by *highly reliable* medical device adapters to enable automated open loop safety protocols is a preferred long term solution. However, a simple near term solution is to have a ***highly reliable MFM***. The MD PnP supervisor should follow the IHM protocol for sending commands to the ventilator and laser devices. The communication between the MD PnP supervisor and the medical devices in airway laser surgical scenario includes the following steps:

- The MD PnP supervisor sends a $pause$ command annotated with "$to\ be\ resumed\ at\ time\ t$" to the MFM and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.
- If an acknowledgment is received, the MD PnP supervisor then sends $pause\ ventilator$ command and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.
- After waiting for T seconds to ensure that the oxygen in patient's airway is cleared, the MD PnP supervisor sends $turn\ on\ laser$ command to the MFM and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.
- If an acknowledgment is received, the MD PnP supervisor sends $turn\ on\ laser$ command to

the device and waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor to let medical staff know that the laser is not responding.

- When pause time limit reaches, the MD PnP supervisor sends $turn\ off\ laser$ command first to the MFM and then to the laser after an acknowledgment is received from the MFM. Next, the supervisor waits for an acknowledgment; if one is not received, an alarm is sounded by the supervisor.

- If an acknowledgment from laser is received, the MD PnP supervisor sends $resume\ ventilator$ command first to the MFM, and then to the ventilator after an acknowledgment is received from the MFM. Next, the supervisor waits for an acknowledgment from the ventilator; if one is not received, an alarm is sounded by the supervisor to let medical staff know that the ventilator is not responding.

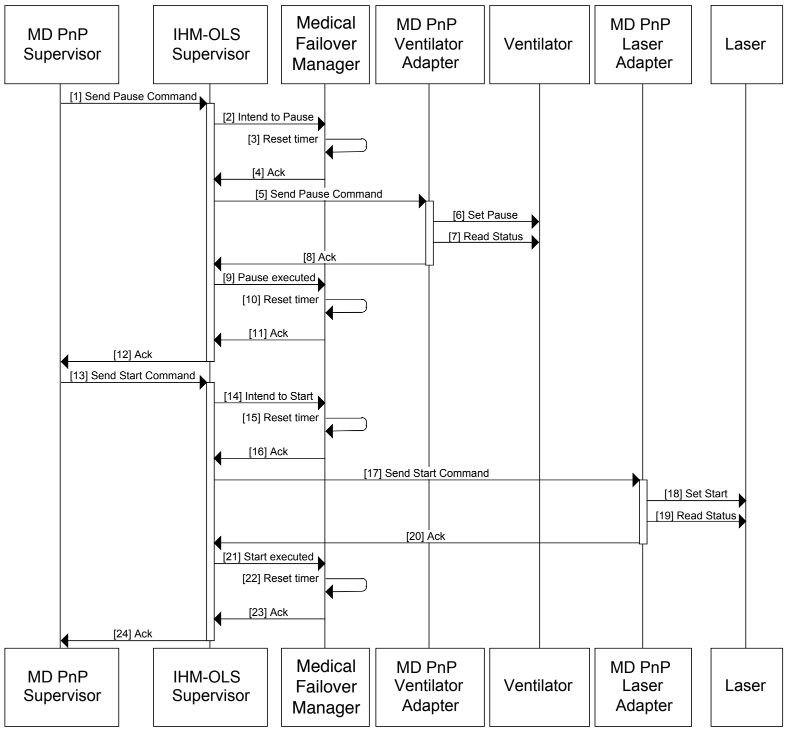
- During the operation, whenever the MFM stops receiving heartbeats from the MD PnP supervisor, an alarm is sounded by the MFM to indicate that the MD PnP supervisory loop is open. In addition, the list of recent commands is displayed.

Figures 6.15(a) and 6.15(b) show the sequence of messages exchanged among the different components of the IHM open-loop safe MD PnP system in the normal and crash operation modes, receptively.

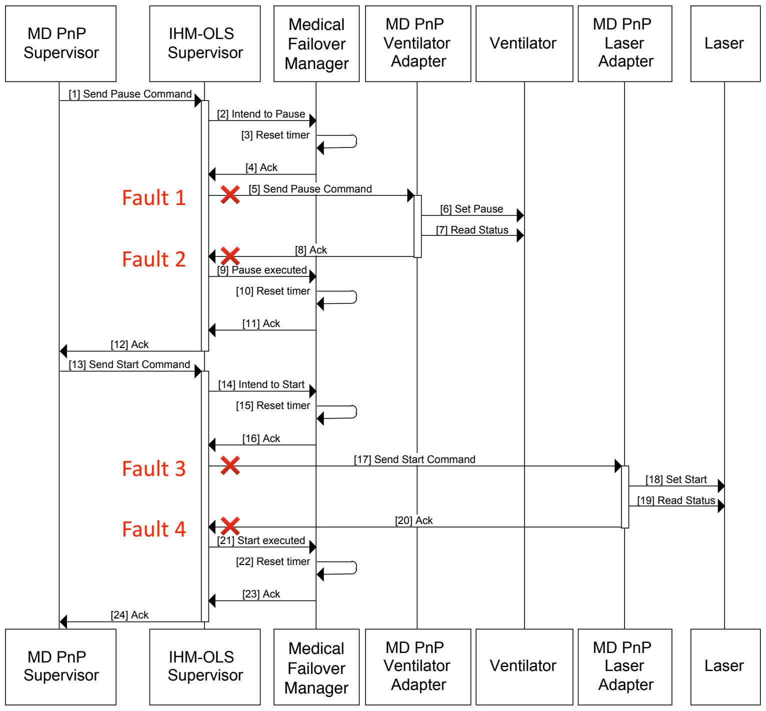
At the last step of our safety analysis procedure, we redraw the extended fault tree for surgical fire (step **S4**). Figure 6.10(a) shows the MD PnP therapeutic configuration subtree when device adapters are only assumed to be ***reliable*** ($Subtree$ 4.1). The subtree for unsafe device configuration in the IHM open-loop safe MD PnP system ($Subtree$ 4.2) is shown in Figure 6.10(b). Figure 6.16 shows the updated surgical fire tree that includes these subtrees. The addition of ***highly reliable MFM*** in $Subtree$ 4.2 and the AND gate on top of MD PnP and IHM open-loop safe subbranches (depicted by dashed blue rectangles in Figure 6.16) results in removal of all paths to the root.

6.4.3 Patient Controlled Analgesia

The main adverse event in the patient controlled analgesia case is morphine overdose. This safety hazard is shown under *Unsafe Therapeutic Configuration* $\rightarrow$ *Unsafe Medication Administration* $\rightarrow$



(a) Normal Mode Operation



(b) Open-Loop Safe Operation

Figure 6.15: IHM Open-Loop Safe Sequence Diagram for Airway-Laser Surgery

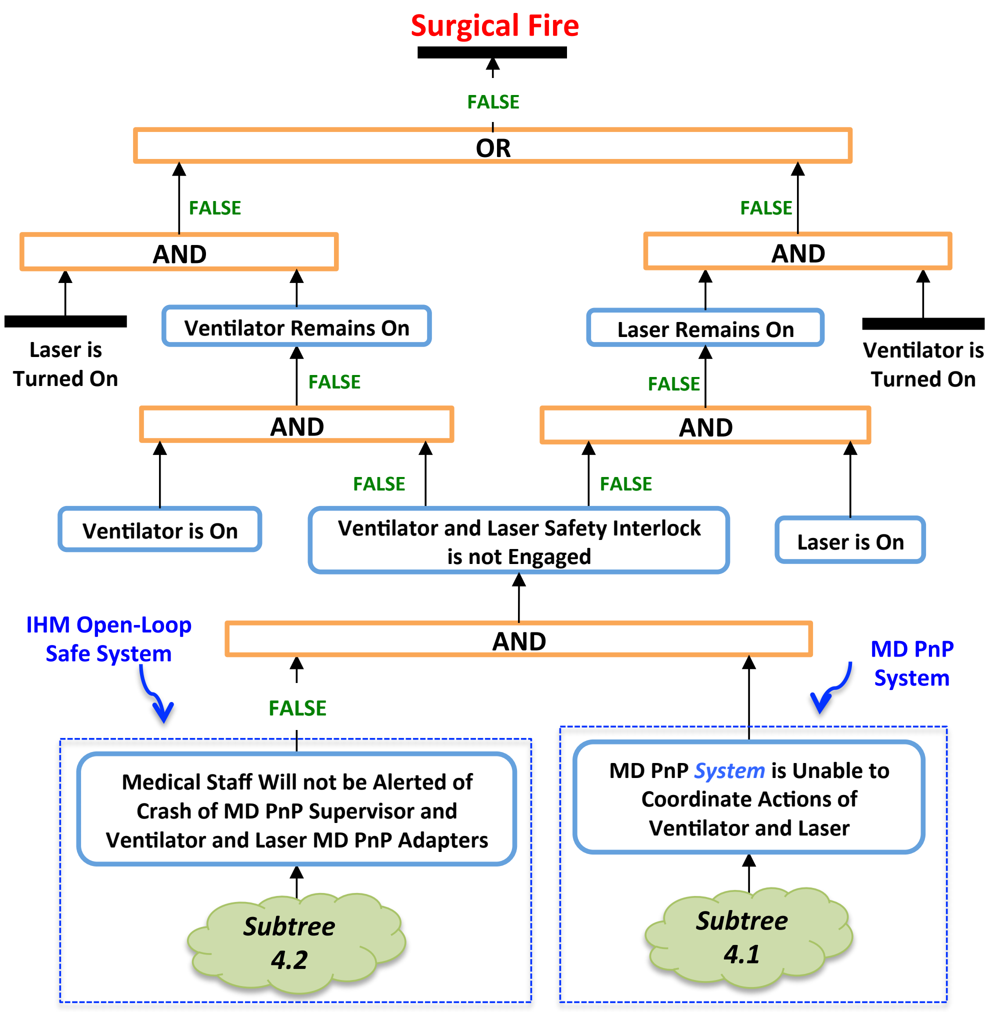

Surgical Fire
FALSE
OR
FALSE
FALSE
AND
AND
Laser is Turned On
Ventilator Remains On
Laser Remains On
Ventilator is Turned On
FALSE
FALSE
AND
AND
FALSE
FALSE
Ventilator is On
Ventilator and Laser Safety Interlock is not Engaged
Laser is On
IHM Open-Loop Safe System
AND
MD PnP System
FALSE
Medical Staff Will not be Alerted of Crash of MD PnP Supervisor and Ventilator and Laser MD PnP Adapters
MD PnP System is Unable to Coordinate Actions of Ventilator and Laser
Subtree 4.2
Subtree 4.1

Figure 6.16: IHM Open-Loop Safe MD PnP System Fault Tree Analysis for Surgical Fire Event

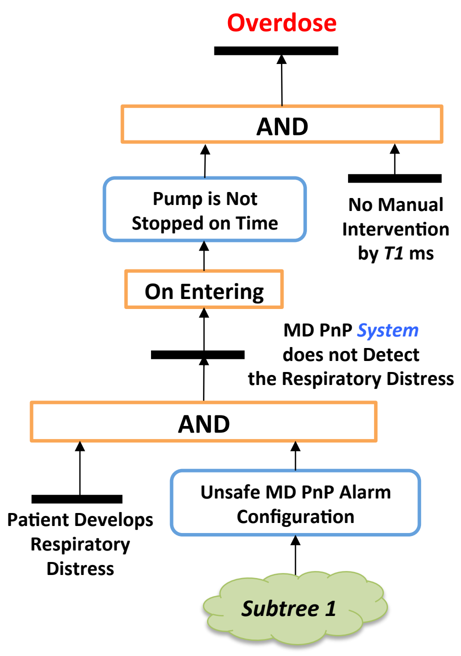
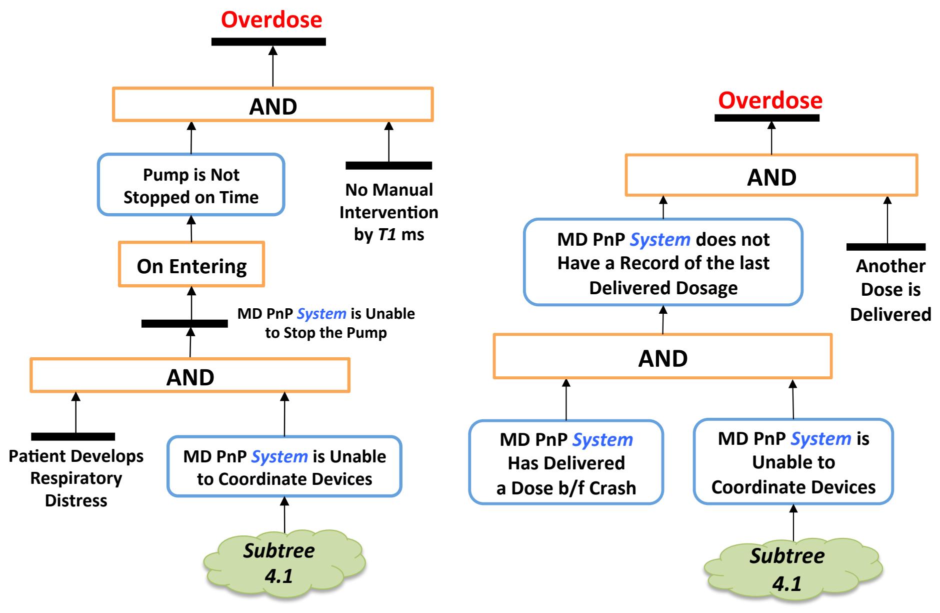

Overdose
AND
Pump is Not Stopped on Time
No Manual Intervention by T1 ms
On Entering
MD PnP System does not Detect the Respiratory Distress
AND
Patient Develops Respiratory Distress
Unsafe MD PnP Alarm Configuration
Subtree 1

Figure 6.17: Extended Fault Tree Analysis of Morphine Overdose as a Result of Unsafe Monitoring Configuration


Overdose
AND
Pump is Not Stopped on Time
No Manual Intervention by T1 ms
On Entering
MD PnP System is Unable to Stop the Pump
AND
Patient Develops Respiratory Distress
MD PnP System is Unable to Coordinate Devices
Subtree 4.1
Overdose
AND
MD PnP System does not Have a Record of the last Delivered Dosage
Another Dose is Delivered
AND
MD PnP System Has Delivered a Dose b/f Crash
MD PnP System is Unable to Coordinate Devices
Subtree 4.1

(a) Case 1 (b) Case 2

Figure 6.18: Extended Fault Tree Analysis of Morphine Overdose as a Result of Unsafe Therapeutic Configuration

Incorrect Bolus Dosage in Figure 6.25. The extended fault trees in this section show how this sub branch is populated in the PCA case study.

The MD PnP system in this case includes a smart alarm application and a close loop control for PCA pump. Failure of the smart alarm application to detect respiratory distress (Figure 6.17) or failure of closed loop control to stop the pump on time (Figure 6.18) will lead to overdose. An open loop safe solution is required to prevent these hazards when the supervisory loop is open. A simple version can be setting a timer to stop the PCA pump when the supervisor or network crashes, before sending the command to start the PCA pump. The timer value can be updated as the new physiological data on patient state becomes available. The idea is similar to the previous cases; the open-loop supervisor sends a sequence of timed actions to the PCA device starting with an action to ensure the PCA is moved to a safe state when the MD PnP supervisor crashes or the network becomes unavailable. This approach will break the paths, shown in Figures 6.17 and 6.18(a), to the root.

Figure 6.18(b) presents the case, in which the MD PnP closed-loop control application fails before receiving an acknowledgment from the PCA device. To break this path to the root we recommend the use of MFM that records the intent to deliver a dosage before actually sending the corresponding command to the PCA device. If this action is not taken and the MD PnP system crashes after delivering a dosage and before receiving the acknowledgment from the device, there would be no record of the dosage administration, which will potentially lead to an overdose. Making a record of intention to deliver a dosage can break this path to the root. In this case, even if the MD PnP supervisor crashes after sending the command to the PCA and before receiving an acknowledgment from the device, the intention to deliver a dosage is already recorded by the MFM.Therefore, when the MD PnP supervisor restarts it can access this recorded information and avoid sending the commend to deliver another dosage.

Note that fault trees include $Subtree\ 4.1$ representing unsafe therapeutic configuration, and $Subtree\ 1$ representing unsafe monitoring configuration in the MD PnP system. We presented $Subtree\ 4.1$ when discussing brain damage and fire fault trees in the first two clinical examples; $Subtree 1$ is introduced later in the report when studying the MD PnP derived alarms.

6.4.4 Prepare Intensive Care Unit to Receive Post-OP Cardiac Patient

The MD PnP supervisor reads the settings and status of medical devices used in the operating room, when preparing an intensive care unit (ICU) for receiving post-op patients. This information includes the medication and infusion rate for all IV pumps, the setting of mechanical ventilation, the number of invasive pressure channels being monitored, etc. A simple handshaking process is recommended to ensure that complete list of devices and corresponding setting is received by the MD PnP supervisor in the intensive care unit. In the next step, all ICU medical devices are preset to OR settings by the MD PnP supervisory application. The clinician confirms all automatic device settings and enters the clinical data that cannot be retrieved automatically.

The MD PnP middleware must have a communication component to handle remotely accessing medical devices connected to clinical network. Devices such as ventilator have different modes of operation and settings. The translation and mapping of setting from one device to another are performed by the interoperability component of the MD PnP middleware and must be verified as clinically correct.

We assume all medical devices use the same interface description language. The MD PnP middleware can then provide a uniform format to clinical application designers for interacting with medical devices. The middleware translates the uniform format to a specific device setting format. Therefore, the complexity of heterogeneous devices will be hidden from the applications and encapsulated by the MD PnP middleware. As explained above, we assume that the correctness of the MD PnP interoperability logic is verified. Figure 6.19 shows the overall architecture of the MD PnP framework for the clinical scenario involving preparation of ICU to receive post operation patients. The device configuration setting must be verified by human before being applied to the devices in ICU and therefore no device is directly manipulated by the MD PnP components. Therefore, there is no causal path from the MD PnP components to potential safety hazards in this clinical scenario.

6.4.5 Home to Hospital

The data coming from telehealth (TH) monitoring devices, used at home, may be unreliable or inaccurate. Inaccurate physiological measurements from these devices may be used in addition to more accurate data from hospital devices to monitor patient state. However, this may result in unintentional

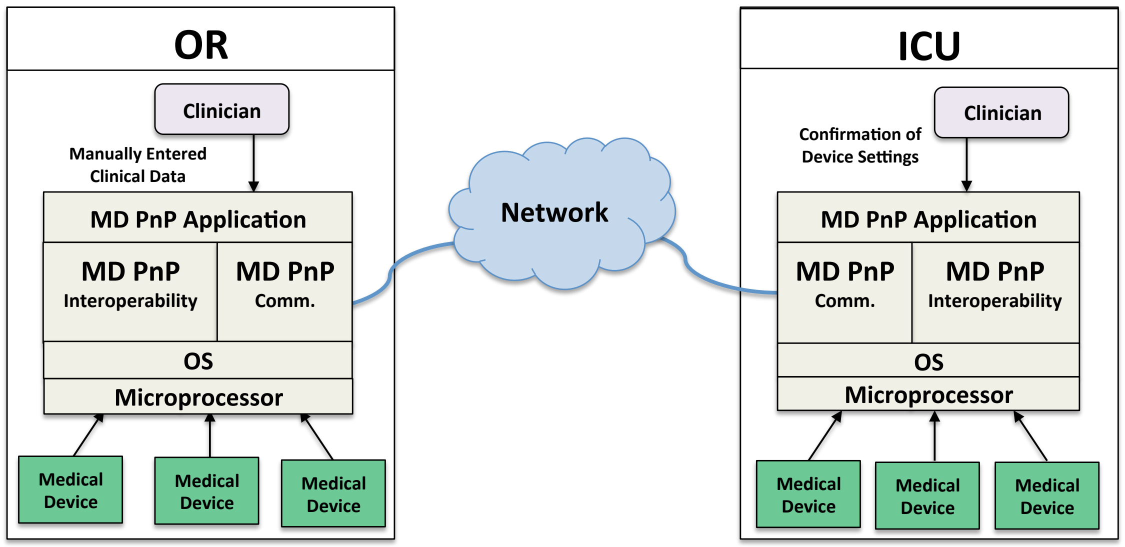


Figure 6.19: The MD PnP System for Preparing ICU to Receive Post-OP Cardiac Patient

inaccuracy in patient state assessment. Properly managing the heterogeneity of medical sensors' reliability can alleviate this problem. One approach is exporting the information regarding the reliability of different medical sensors to the application layer. Considering the accuracy of different medical sensor readings when making decisions about patient state can increase the safety and effectiveness of the patient monitoring process.

The MD PnP interoperability program recommends a unified model for medical sensors and physiological measurements, which is defined as an abstraction of device and measurement information relevant to the safety and effectiveness of patient care, including the accuracy of measurements. One such model is shown in Table 6.4. Most of the chosen attributes for our measurement model are self-describing. Some sensor devices generate more than one physiological measurement with different accuracies. Therefore, accuracy information must be included for each pair of device and physiological measurement. We show examples of the measurement model instantiation; the values stored in the model instants are only nominal values. The fields can be populated as the information becomes available.

Physiology	Monitor Type	Device ID	Patient ID	Accuracy	Current Reading
Blood Pressure	Blood Pressure Cuff	BP-ICU211	1236780	90%	80 mmHg
SpO_2	Oximeter	BP-Cuff-ICU211	1236780	97%	95%
Pulse Rate	Oximeter	Oximeter-ICU211	1234568	95%	100 bpm
Heart Rate	EKG-TH	EKG-1232211	1232211	95%	95 bpm
Heart Rhythm	EKG	EKG-ICU211	1236780	98%	Sinus
$ScvO_2$	Central Venous Catheter	CVC-ICU211	1236780	99%	80%
CVP	Central Venous Catheter	CVC-ICU211	1236780	99%	80 mmHg

Table 6.4: Unified Device and Measurement Models

6.4.6 IV Sedation for a GI Procedure

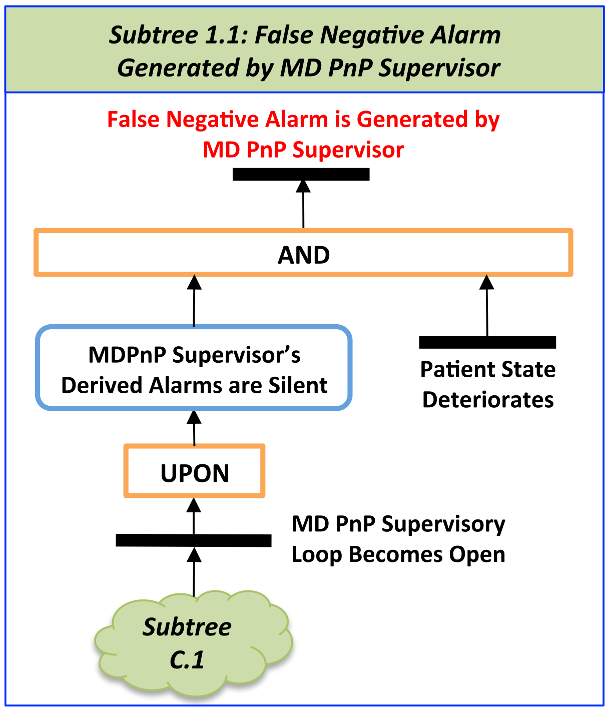
The MD PnP framework aims at improving the safety and effectiveness of patient care by integrating multiple devices for additional intelligence. For this clinical scenario, there are four major components involved in creating a safer clinical environment with inter-operable medical devices. We look at each component and analyze whether they can contribute to any of the known potential safety hazards.

Virtual display

The goal here is to have one integrated main display, which receives the clinical data from different monitoring devices. For this scenario, SpO_2, $ETCO_2$ (end-tidal CO_2), readings from a depth of anesthesia monitor, respiratory rate and blood pressure can be displayed remotely on a central monitor. However, the crash failure detection is simple and straightforward for a display. If the virtual display crashes remote monitoring would not be possible; however, the clinical data are still displayed on individual monitoring devices. Therefore, as long as it is verified that the virtual display will not alter any value it receives from monitoring devices, this component will not cause any safety hazards.

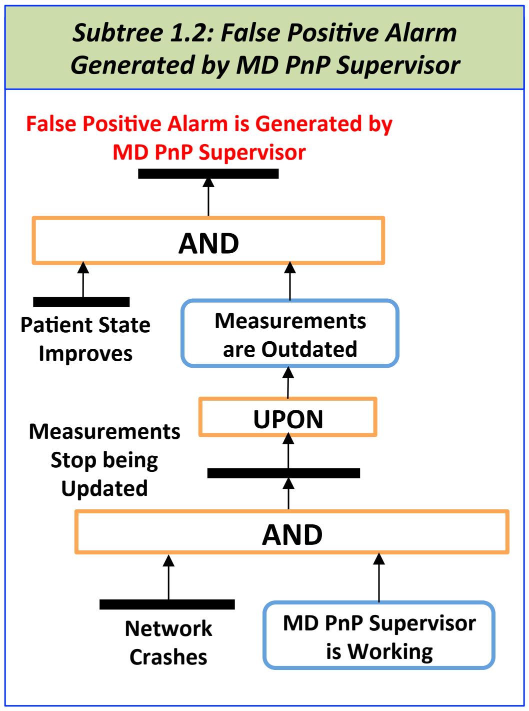
Derived alarms

The main safety hazards for an MD PnP derived alarm system are 1) false negative alarms 2) false positive alarms and 3) overlooked true positive alarms, which are the final result of false positive alarms. Figures 6.20(a), 6.20(b), and 6.21 show the extended fault trees for alarm hazards. We name these three



Subtree 1.1: False Negative Alarm Generated by MD PnP Supervisor
False Negative Alarm is Generated by MD PnP Supervisor
AND
MDPnP Supervisor's Derived Alarms are Silent
Patient State Deteriorates
UPON
MD PnP Supervisory Loop Becomes Open
Subtree C.1

(a) False Negative



Subtree 1.2: False Positive Alarm Generated by MD PnP Supervisor
False Positive Alarm is Generated by MD PnP Supervisor
AND
Patient State Improves
Measurements are Outdated
UPON
Measurements Stop being Updated
AND
Network Crashes
MD PnP Supervisor is Working

(b) False Positive

Figure 6.20: False Alarm Subtrees for MD PnP Derived Alarms

$Subtree$ 1.1, $Subtree$ 1.2, and $Subtree$ 1.3, respectively. Using these subtrees, we draw the extended fault tree for unsafe monitoring configuration in the MD PnP system (Figure 6.22).

To prevent failure of the MD PnP components from contributing to alarm safety hazards, some precaution must be taken. The generic solution is to use MFM and take the IHM based approach, in which:

- The MD PnP supervisor queries each device periodically.
- If a device fails to respond for n consecutive cycles, the supervisor sounds an alarm to notify medical staff that device X has failed to communicate
- The MD PnP supervisor sends periodic heart beats to the *highly reliable* MFM.
- If the MFM fails to receive heartbeat from the MD PnP supervisor, then MFM sounds a local alarm indicating that the supervisor has failed to respond and displays the recent commands.

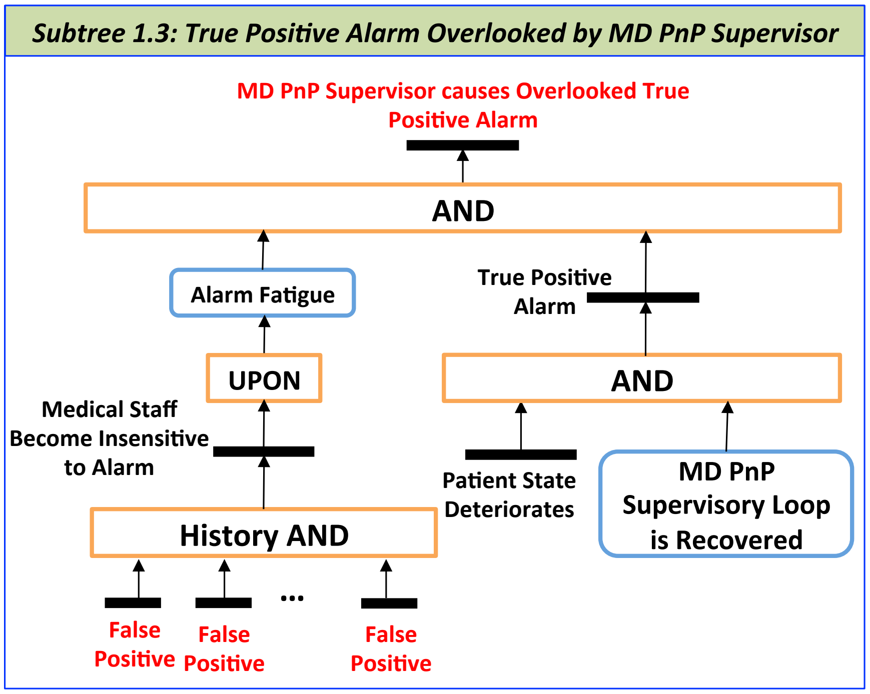

Subtree 1.3: True Positive Alarm Overlooked by MD PnP Supervisor
MD PnP Supervisor causes Overlooked True Positive Alarm
AND
Alarm Fatigue
True Positive Alarm
UPON
AND
Medical Staff Become Insensitive to Alarm
Patient State Deteriorates
MD PnP Supervisory Loop is Recovered
History AND
...
False Positive
False Positive
False Positive

Figure 6.21: Overlooked True Positive Subtree for MD PnP Derived Alarms

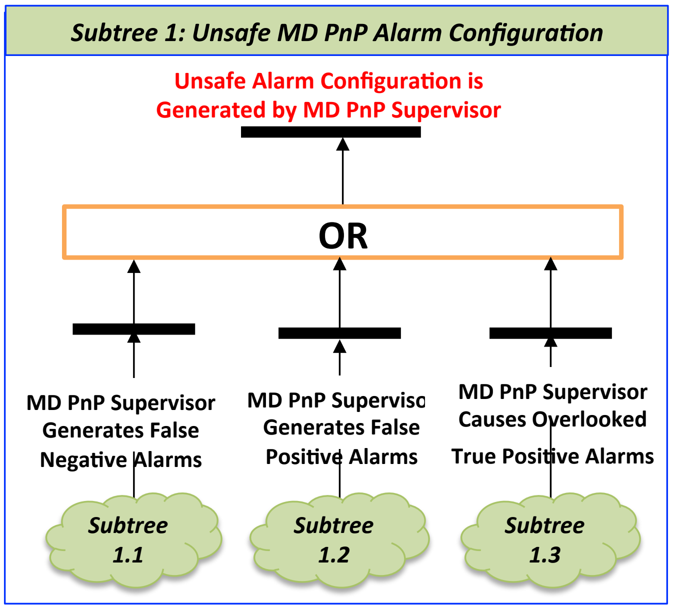

Subtree 1: Unsafe MD PnP Alarm Configuration
Unsafe Alarm Configuration is Generated by MD PnP Supervisor
OR
MD PnP Supervisor Generates False Negative Alarms
MD PnP Supervisor Generates False Positive Alarms
MD PnP Supervisor Causes Overlooked True Positive Alarms
Subtree 1.1
Subtree 1.2
Subtree 1.3

Figure 6.22: Unsafe Monitoring Configuration Subtree for MD PnP Derived Alarms

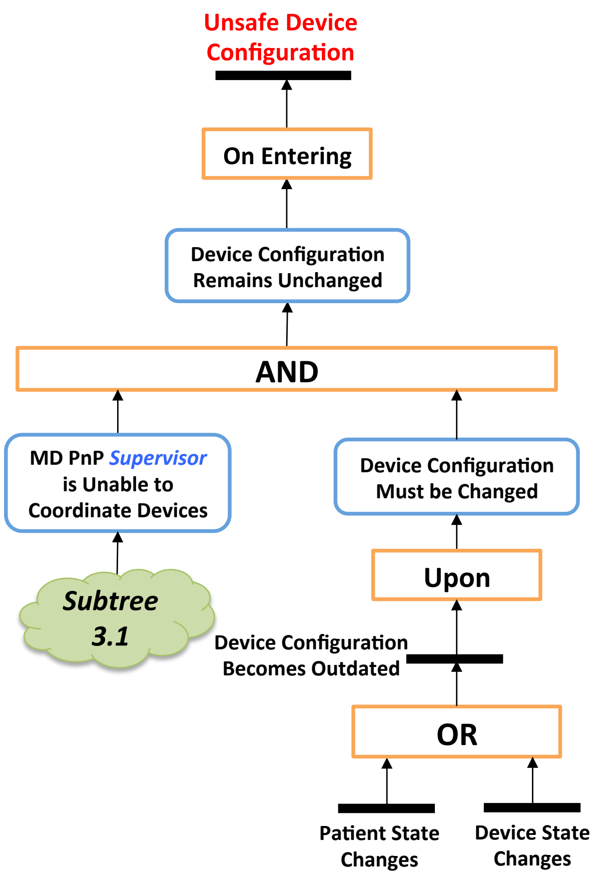

Unsafe Device Configuration
On Entering
Device Configuration Remains Unchanged
AND
MD PnP *Supervisor* is Unable to Coordinate Devices
Device Configuration Must be Changed
Subtree 3.1
Upon
Device Configuration Becomes Outdated
OR
Patient State Changes
Device State Changes

Figure 6.23: Extended Fault Tree Analysis for the MD PnP Virtual Front Panel

- Medical staff must enable all the local alarms in individual devices and restart the MD PnP supervisor and go through all the safety checks.

Virtual front panel/ remote configuration

The extended fault tree for this component is shown in Figure 6.23. The root of this tree is unsafe device configuration, which is remotely generated by this MD PnP component. This may include incorrect setting of therapeutic devices such as ventilator, unsafe IV medication dosage, or incorrect cutoff threshold values for monitoring devices. Therefore, it is critical to detect crash of the MD PnP supervisory loop, using methods such as IHM protocol along with MFM. When crash of MD PnP components is detected by MFM, a local alarm must be sounded to get the attention of medical staff. The alarm would indicate that the remote front panel has crashed and the devices must be locally set by medical staff.

Autonomous control

The autonomous control component is responsible for implementing safety interlocks and physiological closed loop controls. Verifying the logical correctness of this component is critical. The safety interlock can prevent unsafe interaction between devices and treatment actions. One example is the safety interlock between an oxygen controller and surgical laser in the airway laser surgical scenario. Absence of such interlocks can directly endanger patient safety. In the case of closed loop control, settings of therapeutic devices and IV medication are automatically controlled based on the current readings from medical sensors. Therefore, incorrect control values can lead to unsafe therapeutic configuration.

The execution of safety interlock and closed loop control steps can be interrupted when MD PnP platform crashes and as shown in Figure 6.24 may lead to unsafe device operation. The near term generic solution is to use the IHM open loop safety approach. Each step of the interlock protocol must have a timed acknowledgment or an alarm will be raised by the MD PnP supervisor, which is in turn monitored by the ***highly reliable MFM***. However, it shall be user's responsibility to verify the interlock protocol together with IHM open loop safety approach, because currently there is no known general proof that the generic open-loop safety protocol will be compatible with all possible interlock logics.

6.5 A Generic Approach for Safety Analysis of the MD PnP Systems

In this section we look at the steps of a safety analysis procedure for the generic MD PnP system. We have looked at examples of potential safety hazards in clinical environments caused by incorrect and unsafe medical device configuration. Figure 6.25 presents a general categorization of such potential safety hazards. We consider three main sources of safety hazards: 1) unsafe monitoring configuration, 2) unsafe therapeutic configuration, and 3) unsafe interaction between monitoring and therapeutic configurations. We have populated a subset of branches to demonstrate the general hierarchy. The brain damage hazard explained in the previous section is an example of a hazard caused by an unsafe therapeutic configuration, specifically incorrect ventilator device setting. An example of interaction hazard

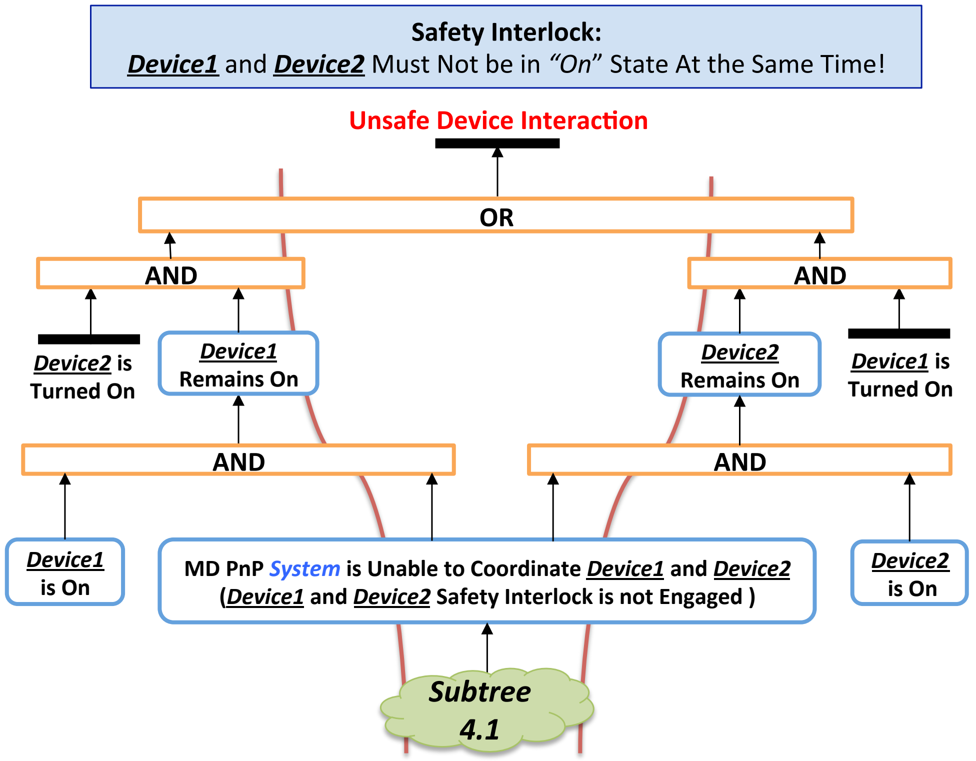

Safety Interlock:
Device1 and Device2 Must Not be in "On" State At the Same Time!
Unsafe Device Interaction
OR
AND
AND
Device2 is Turned On
Device1 Remains On
Device2 Remains On
Device1 is Turned On
AND
AND
Device1 is On
MD PnP System is Unable to Coordinate Device1 and Device2 (Device1 and Device2 Safety Interlock is not Engaged)
Device2 is On
Subtree 4.1

Figure 6.24: Extended Fault Tree Analysis for the MD PnP Autonomous Control Systems

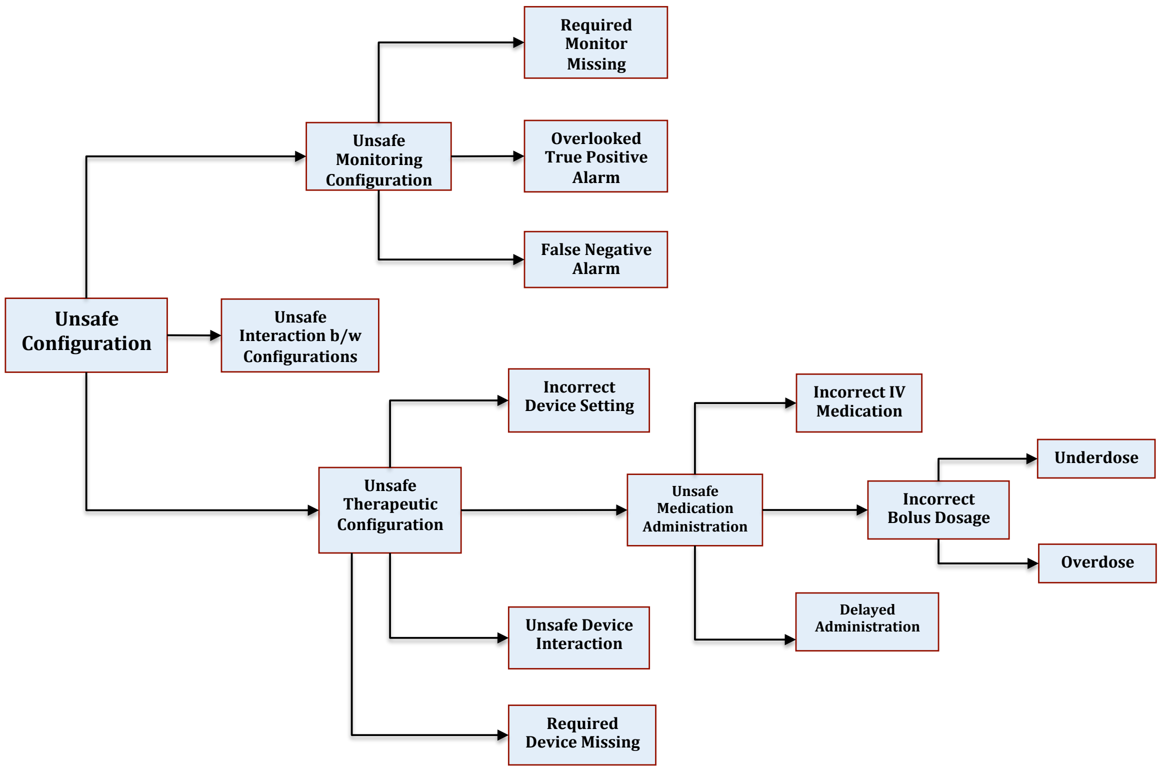


Figure 6.25: A Generic Categorization of Potential Safety Hazards

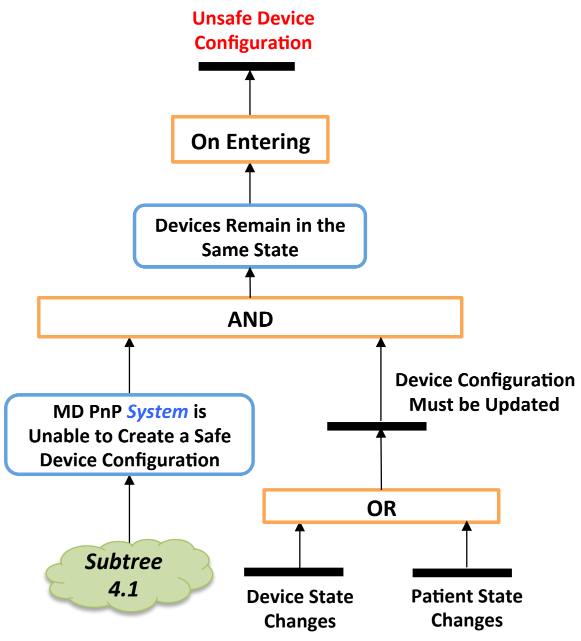

Unsafe Device Configuration
On Entering
Devices Remain in the Same State
AND
MD PnP System is Unable to Create a Safe Device Configuration
Device Configuration Must be Updated
OR
Subtree 4.1
Device State Changes
Patient State Changes

Figure 6.26: Extended Fault Tree Analysis of the MD PnP System with *Reliable* Device Adapters

is the surgical fire hazard that is caused by unsafe interaction of two therapeutic devices: ventilator and surgical laser.

Step1: For each clinical scenario, enumerate the potential safety hazards and study the *unsafe configuration* causing each hazard. As described in the previous section, use the extended fault tree to demonstrate the system behavior leading to hazardous situations. Figure 6.26 shows a generic extended fault tree for the MD PnP system with ***reliable*** device adapters. The tree includes $Subtree$ 4.1, which was presented in Figure 6.10(a).

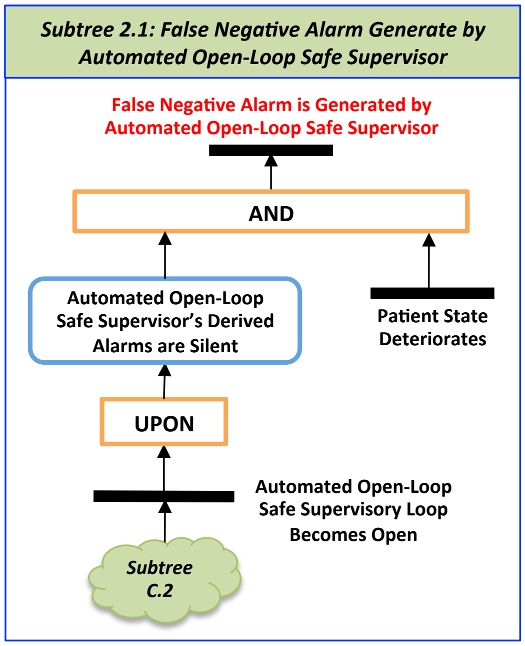
Step 2: Determine if there is a causal chain from the MD PnP components in the system to any unsafe configuration causing the safety hazard. The root of this tree, indicated by *Unsafe Device Configuration*, encompasses different safety hazards caused by unsafe device configuration. For each clinical scenario, detailed information about specific device configuration and safety hazards substitutes for this term in the generic tree. As presented in the example scenarios, there are potential paths from the MD PnP components to unsafe configuration which ultimately results in a safety hazard. Note that this is true for clinical scenarios, in which the medical devices are directly manipulated by the MD PnP components. For example, in airway laser surgical scenario the MD PnP supervisor controls and coordinates the actions to turn on, turn off and change the settings of ventilator (oxygen supply) and surgical laser. Therefore, failure of any MD PnP component potentially leads to an unsafe device configuration and ultimately surgical fire or brain damage.

For clinical cases in which the medical devices are directly manipulated by the MD PnP components, we use an open-loop safe approach to prevent the failure of the MD PnP components from contributing to potential safety hazards. We discussed two different approaches: *Automated Open-Loop Safety* and *IHM Open-Loop Safety*.

Step 3 & 4 for Automated Open-Loop Safe Approach: Recommend an automated open-loop safe solution and redraw the extended fault tree for each safety hazard. In the automated protocol, we require ***highly reliable*** device adapters. A set of commands, including timed safety actions, are sequentially sent to medical devices. At each step, we wait for an acknowledgment from the device before sending the subsequent commands. When sending a command to move a device into a state, which is only temporarily safe, first a safety timed action is sent to the device. This action will be taken by the device if the MD PnP supervisory loop becomes open, and will move the device to a safe state after the specified time elapses. For example, in the laparoscopic cholecystectomy scenario, a timed safety action, indicating that the ventilator must be resumed after a specific time period, is sent to the ventilator. When the device acknowledges that the timer to resume the ventilator is set up, the MD PnP supervisor sends the command to turn off the ventilator. Note that moving the ventilator to the off state is only temporarily safe; therefore, the open loop safe protocol takes the safety precautions before sending the corresponding command. Refer to Figures 6.7 and 6.13 for of message sequence communicated between different components of the automated open-loop safe MD PnP system. This approach ensures that the failure of the supervisory loop at any step during the communication will not leave the system in an unsafe configuration, given the initial device configuration is safe. The main components of the automated open-loop safe MD PnP system is shown in Figure 6.3.

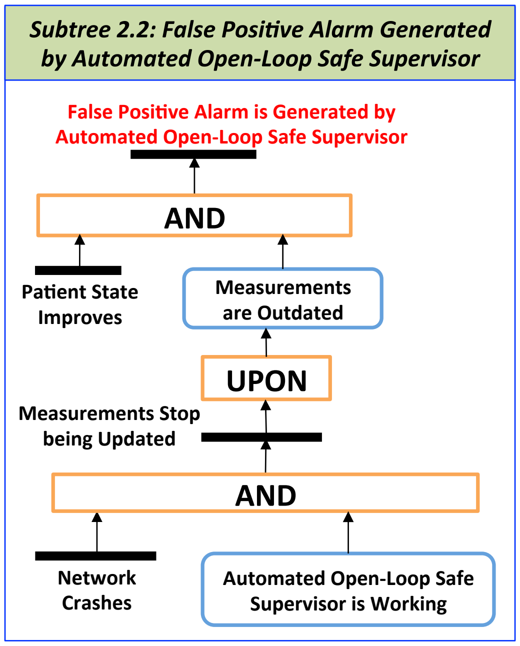
At the last step of safety analysis procedure, we redraw the extended fault tree for each safety hazard. The therapeutic configuration subtree for automated open-loop safe system, $Subtree$ 3.2, was presented in Figure 6.8(a). Figures 6.27(a), 6.27(b), and 6.27(c) show the extended fault trees for alarm hazards. We name these three $Subtree$ 2.1, $Subtree$ 2.2, and $Subtree$ 2.3, respectively. Using these subtrees, we draw $Subtree$ 2 for unsafe monitoring configuration in the automated open-loop safe MD PnP system (Figure 6.28). Putting together the monitoring and therapeutic subtrees, we have the general extended fault tree fort automated open-loop safe MD PnP system in Figure 6.29. As presented in $Subtree$ 3.2 and $Subtree$ 2, the automated open loop safe supervisor may fail. However, the open loop safe protocol is *any-step safe*; the open-loop safe supervisor always ensures that the MD PnP device has received the timed safety action before moving to a transient safe state. Moreover, the device adapters are assumed to be ***highly reliable*** devices. Therefore, if the supervisory loop becomes open, the adapter will move the device into a safe state. In conclusion, assuming the initial configuration is safe, failure of the supervisory loop at any step of the handshaking protocol will not result in unsafe ventilator configuration. Finally, the addition of AND gate on top of MD PnP and open-loop safe subbranches (depicted by dashed blue rectangles in Figure 6.29) results in removal of all paths to the root.

Step 3 & 4 for IHM Open-Loop Safe Approach: Recommend an integrated human-machine open-loop safe solution and redraw the extended fault tree for each safety hazard. The main components of an IHM-based open-loop safe MD PnP system are shown in figure 6.4. The MFM must be ***highly reliable***, and should have its own power supply and a built-in alarm. The interface between timer and



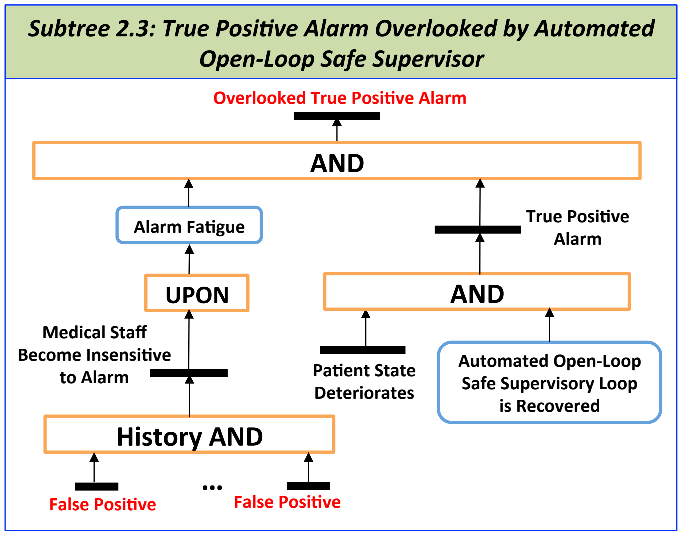
Subtree 2.1: False Negative Alarm Generate by Automated Open-Loop Safe Supervisor
False Negative Alarm is Generated by Automated Open-Loop Safe Supervisor
AND
Automated Open-Loop Safe Supervisor's Derived Alarms are Silent
Patient State Deteriorates
UPON
Automated Open-Loop Safe Supervisory Loop Becomes Open
Subtree C.2

(a) False Negative Alarm



Subtree 2.2: False Positive Alarm Generated by Automated Open-Loop Safe Supervisor
False Positive Alarm is Generated by Automated Open-Loop Safe Supervisor
AND
Patient State Improves
Measurements are Outdated
UPON
Measurements Stop being Updated
AND
Network Crashes
Automated Open-Loop Safe Supervisor is Working

(b) False Positive Alarm



Subtree 2.3: True Positive Alarm Overlooked by Automated Open-Loop Safe Supervisor
Overlooked True Positive Alarm
AND
Alarm Fatigue
True Positive Alarm
UPON
AND
Medical Staff Become Insensitive to Alarm
Patient State Deteriorates
Automated Open-Loop Safe Supervisory Loop is Recovered
History AND
False Positive
...
False Positive

(c) Overlooked True Positive

Figure 6.27: Alarm Hazard Subtrees for Automated Open-loop Safe MD PnP System

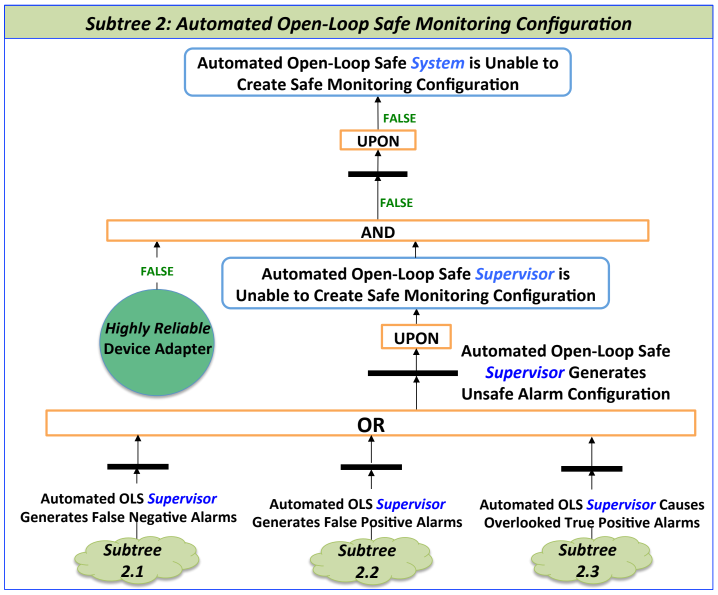

Subtree 2: Automated Open-Loop Safe Monitoring Configuration
Automated Open-Loop Safe System is Unable to Create Safe Monitoring Configuration
FALSE
UPON
FALSE
AND
FALSE
Automated Open-Loop Safe Supervisor is Unable to Create Safe Monitoring Configuration
Highly Reliable Device Adapter
UPON
Automated Open-Loop Safe Supervisor Generates Unsafe Alarm Configuration
OR
Automated OLS Supervisor Generates False Negative Alarms
Automated OLS Supervisor Generates False Positive Alarms
Automated OLS Supervisor Causes Overlooked True Positive Alarms
Subtree 2.1
Subtree 2.2
Subtree 2.3

Figure 6.28: Monitoring Configuration Subtree of Automated Open-loop Safe MD PnP System

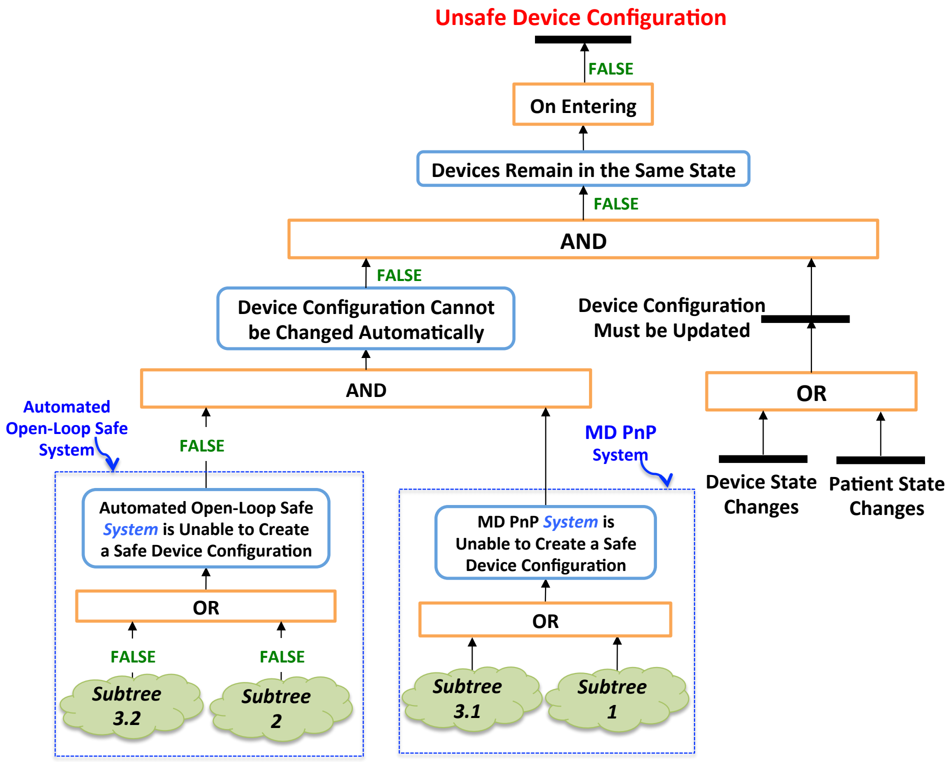
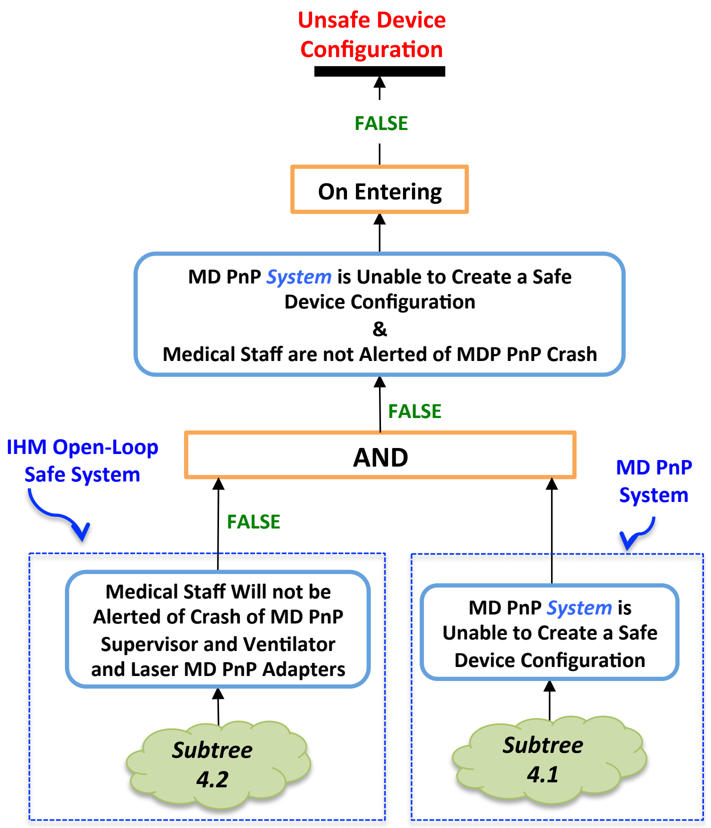

Unsafe Device Configuration
FALSE
On Entering
Devices Remain in the Same State
FALSE
AND
FALSE
Device Configuration Cannot be Changed Automatically
Device Configuration Must be Updated
AND
OR
Automated Open-Loop Safe System
FALSE
MD PnP System
Device State Changes
Patient State Changes
Automated Open-Loop Safe System is Unable to Create a Safe Device Configuration
MD PnP System is Unable to Create a Safe Device Configuration
OR
OR
FALSE
FALSE
Subtree 3.2
Subtree 2
Subtree 3.1
Subtree 1

Figure 6.29: Extended Fault Tree Analysis of Automated Open-loop Safe MD PnP System



Unsafe Device Configuration
FALSE
On Entering
MD PnP System is Unable to Create a Safe Device Configuration & Medical Staff are not Alerted of MDP PnP Crash
FALSE
AND
IHM Open-Loop Safe System
MD PnP System
FALSE
Medical Staff Will not be Alerted of Crash of MD PnP Supervisor and Ventilator and Laser MD PnP Adapters
MD PnP System is Unable to Create a Safe Device Configuration
Subtree 4.2
Subtree 4.1

Figure 6.30: Extended Fault Tree Analysis of IHM Open-loop Safe MD PnP System

the MD PnP supervisor should be wired; USB port is one example. The medical device adapters, in this design, only need to be cleared as ***reliable***. Figures 6.9(a) and 6.15(a) show the sequence of messages communicated between the different components of IHM open-loop safe MD PnP system for the first two clinical examples.

At the last step, we redraw the fault tree for each safety hazard considering the IHM open-loop safety approach in place. The subtrees for MD PnP system with ***reliable*** device adapters ($Subtree$ 4.1) and IHM open-loop safe system ($Subtree$ 4.2) have been presented in Figures 6.10(a) and 6.10(b), respectively. Figure 6.30 shows the general extended fault tree for the IHM open-loop safe MD PnP system, drawn using these subtrees. The addition of ***highly reliable MFM*** and the AND gate on top of MD PnP and IHM open-loop safe subbranches (depicted by dashed blue rectangles in Figure 6.30) results in removal of all paths to the root.

6.6 Summary

As noted on the Medical Device Plug-and-Play (MD PnP) program's website, adoption of medical device interoperability opens the door to significantly mitigate the preventable medical errors. The goal of this program is to develop the technological foundation to design the next generation medical systems. In an MD PnP system, medical devices are plugged in as needed, information from different sources are fused, and medical actions are coordinated by providing an integrated clinical environment (ICE).

On the other hand, the MD PnP middleware and applications run on top of commercial computing and communication platforms. There are no approved platforms by any agency that support plug and play. This is because plug and play could generate new configurations not considered during the approval process. Furthermore, all existing commercial platforms include a disclaimer indicating that these platforms are not designed to be used for safety critical systems and applications. Therefore, we need to consider its failure modes and the safety impacts on intended medical applications.

This document provides the guidelines to assist users to safely use the MD PnP system, which typically consists of a commercial platform not designed for safety critical applications. We have illustrated how to determine if the planned MD PnP system usage is a causal factor in the known safety hazards of a medical procedure. We used extended fault tree for safety analysis of MD PnP systems. The need for safety engineering expertise and domain knowledge is known as a limiting factor in the use of fault tree analysis. Ensuring comprehensiveness of fault trees is a challenging task. In complex systems, it is possible to miss some system behavior, which can contribute to the safety hazards not presented in the analysis but exists in real world systems. Thus, for safety critical MD PnP system, the extended fault tree analysis should be approved.

Should the MD PnP usage be a causal factor of safety hazards, we provided guidelines to transform the configuration so that it is no longer a causal factor. We discussed long term and near term solutions:

- The long solution requires device makers to support simple timed safety actions.
- The near term solutions for existing medical devices is to use a Medical Failover Manager (*MFM*) to remind users to perform the timed safety actions.

When the MD PnP system's output are used to manipulate safety critical medical devices, we recommend using MFM and following the IHM handshaking protocol, even when the MD PnP sends safety critical medical devices commands, which are first reviewed and approved by medical staff. Keeping an accurate record for what the MD PnP supervisor has done will be invaluable for post operation analysis, especially settling disputes if any. Hence, MFM's record system should be protected from tampering by unauthorized users.

6.7 Research Publications

The chapter is being prepared for submission to *Journal of Medical Systems*:

- **M. Rahmaniheris**, L. Sha, and Sandy Weininger "*A Framework for the Safe Use of MD PnP Systems*"

The following is an earlier publication of our research on safety of MD PnP systems, which are developed using commercial computation and communication equipments:

- C. Kim, M. Sun, **M. Rahmaniheris** and L. Sha, "*How to Reliably Integrate Medical Devices Over Wireless Network*", 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 [93]

Chapter 7

Discussion & Future Work

In this thesis, we have proposed an organ-centric pathophysiology-based modeling paradigm, where we translate the medical text into executable interactive disease and organ automata. This is possible because disease and organ states are already discretized by the medical community, for example, different stages of cancer, diabetes, heart diseases, etc. We model the patient state and different disease stages according to pathophysiology of organ systems, which describes how each of the organ functions are affected by the disease. Furthermore, the diagnosis and treatment are organized for each stage. We studied two applications: cardiac arrest and sepsis. The developed models are then integrated into a model-driven best practice guidance system. We have conducted a preliminary clinical evaluation in an ACLS training class at Carle hospital, to validate the clinical correctness and usefulness of the cardiac arrest models and our model-driven resuscitation guidance system.

Many of our major clinical requirements originate from an important subset of IOM recommendations [2, 94]. According to the feedbacks from domain experts, we believe that the proposed system can reduce preventable medical errors by addressing one of the important recommendations by this institute: "*avoid reliance on memory and vigilance*" (p.170 [2]). Furthermore, we believe our system can help with adherence to best practice monitoring and treatment plans. This specifically addresses one of the common errors made in resuscitation of cardiac arrest, *failure to anticipate the next step* [61].

The proposed medical knowledge representation and modeling methodology must be able to model different diseases in acute care. Initially, we have used the cardiac arrest case study when developing our model. However, to evaluate the applicability of our proposed modeling paradigm, different diseases or acute care scenarios must be considered. A more complex case study such as ***sepsis***, which is a life-threatening organ dysfunction caused by a regulated host response to infection, can be extremely useful in evaluating the applicability of modeling paradigm. It should be noted that we are collaborating with Carle hospital ICU on the development of a best practice sepsis guidance system.

Despite all the capabilities and potentials of our proposed modeling paradigm, there are several important limitations that deserve consideration. They may seem to be out of our scope; nonetheless, we believe discussing these issues may provide the reader with a more clear understanding of scope and focus of this thesis. Furthermore, we plan on addressing some of the issues as part of our future work.

7.1 Model's Level of Fidelity

The models in stateflow and UPPAAL are very abstract and do not cover all physiological dynamics of the organ function, we need to consider how they can track the organ state with sufficient fidelity. Following that, we must explain how we determine "*What is sufficient fidelity?*". Furthermore, if some organs are modeled with more detail and others with cumulative/aggregate state models, one might wonder how this multi-scale modeling does not lose the necessary fidelity and how it allows us to focus on the primary organs.

The models developed according to our current approach, are used to improve compliance with best practice. To reach this goal, the model must remind the medical staff of the best practice monitoring and treatment guidelines needs to be performed as the patient state changes. Therefore, *we define the model level of fidelity as the degree to which the model can help encode and track the critical changes in physiological functions of different organ systems that physician needs to know so he/she can follow best practice monitoring and treatment guidelines.* We determine the sufficient fidelity, based on physician's input and best practice literature for each case study. To ensure the sufficient fidelity of the model, when constructing the models we apply a set of development guidelines, such as the following:

- If the physician only wants to know about the degree of renal insufficiency, we aggregate the information regarding the kidney pathophysiology as degree of insufficiency

- Different concurrent machines are used to track simultaneous changes
- Actionable states allows us to encode the critical changes that require medical staff to perform a new action

We then validate the sufficient fidelity of the models through clinical simulation, which allows our physician collaborators to validates the clinical (fidelity) requirements against the simulated models.

7.2 Variability and Uncertainty

An important challenge in the domain of clinical modeling is accounting for variability of pathophysiological processes (and observed signals in terms of rates/thresholds) both within the same patient (at different times of the day, under medication, etc.) and across the patient population. Such variability must be expressed (in terms of the signals, thresholds and distributions of rates) and the models must be calibrated to capture pathophysiological states with some confidence. This is not an easy question, however, we need to discuss how variability, and uncertainty in general, can be handled throughout the model.

The current approach only partially addresses this challenge by allowing the user to configure the cutoff thresholds in the beginning. However, additional information are most likely required to account for the variability among different patients as well the same patient in different contexts. Furthermore, the information required to determine these values might not be available at configuration time.

We believe this issue can be addressed effectively by developing *parametric models*. The variability mostly affects the definition of transitions between different pathophysiological states. The fixed state transition guards can be substituted with functions of patient-specific parameters, to cover the variability across the patient population, or more fine-grained patient state parameters, to cover the variability for the same patient. The parameters may include age, gender, existing chronic diseases, medications, treatment history for the past *n* hours, etc. At the beginning, the system can ask the attending physician to enter the values for these parameters; throughout the care process they can be updated as new patient information becomes available.

The choice of parameters and function definitions (how each of these parameters affect the transition guards) are clinical knowledge. The model can encode them, only if such knowledge exist. They can be extracted from best practice or provided by physicians. Additionally, calibration, that is systematic adjustment of parameter choices and transition function definitions, can be performed offline with the help of our physician collaborators. However, note that t*he goal of our models is not improving the fidelity of pathophysiological process or calibrating them.* We just encode the available medical knowledge.

7.3 Hysteresis in the Model

The current models detect changes in pathophysiological states when the corresponding state variables cross some predefined thresholds. However, this may result in hysteresis, that is the lagging of an effect (change in physiological measurements) behind its cause (organ insufficiency), in the model.

Note that predictive models is out of our scope; therefore, we limit ourselves to quantifying the delay distribution from the state change to the observed signal crossing a threshold, for the case studies. Using the available time kinetics of different pathophysiological processes, we can calculate the time elapsed from onset of the insufficiency (when measurement crosses the baseline), to the time the corresponding measurement crosses the predefined threshold in our model.

As an example we use creatinine kinetics in renal insufficiency [95]. Let G be the generation rate of creatinine; C, the serum creatinine concentration; V, the volume of distribution of creatinine ; and K, creatinine clearance. Note that K is different for each patient and for the same patient, it decreases rapidly, as the degree of renal insufficiency grows. The creatinine level can be computed at time t using the following formula:

$C(t) = G/K + [C(0) - G/K] * e^{-Kt/V}$

The time required for serum creatinine to increase from $C(0)$ to $C(t)$ can be described by the following formula:

$t = -V/K * ln[(C(t) - G/K)/(C(0) - G/K)]$

When renal function is normal $K \in [0.87 - 1.39\ dL/min]$. When renal insufficiency develops K can drop to $0.5\ dL/min$ and for severe renal insufficiency can drop below $0.15\ dl/min$. $G \in [0.4 - 2.0$

$dL/min]$ but can decrease when renal insufficiency develops.

Using the above formula we can, for example, estimate the time delay from the point moderate renal insufficiency has developed ($creatinine = 3.8\ mg/dL$) until the threshold in our renal model is crossed ($creatinine = 5.0\ mg/dL$); K is set to 0.15 dL/min; G to 1.5 dL/min, and V to 400 dL:

$t = -400/0.15 * ln[(5.0 - 10.0)/(3.8 - 10.0)] \simeq 530\ minutes \simeq 8\ hours$

7.4 Early Identification of At-Risk Patients

As mentioned above, predictive models are out of scope of this thesis. However, we also believe that developing probabilist organ-centric models can be extremely useful for early identification of at-risk patients. A great example of such use cases is sepsis. Every hour of delay in treatment of septic patients reduces the survival rate by 8% [96]. By adjusting the extent and frequency of monitoring for at-risk patients sepsis can be detected and managed as early as possible. Since frequent monitoring for every patient is not feasible and cost effective, the identification of at-risk patients can reduce the cost of care as well.

We believe using the combination of dynamic Bayesian networks and state machines to represent and track patient state can greatly improve the effectiveness of sepsis screening and management process. Bayesian Network (BN) is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph [97]. Dynamic BN (DBN) is a Bayesian network, which relates variables to each other over adjacent time steps [98]. Several techniques based on BN and DBN have been proposed for screening and management of sepsis [99], [100], [101]. However, they often categorize the patients into septic and non-septic categories. No further information on patient state or monitoring guidelines are provided for the latter category. Furthermore, the networks developed and used in these techniques may become complex and unmanageable as more clinical criteria for sepsis screening is added to improve the prediction performance.

Using state-based Bayesian analysis, we can develop modular Bayesian networks and encode different sub-networks in different pathophysiological states. We then calculate the risk of a patient moving toward an insufficiency state. Combining the risk-based analysis with rate calculation (how fast the patient is deteriorating) the model can provide useful information regarding patient state before the thresholds are actually crossed. The guidelines on extent and frequency of monitoring can also be encoded in the states. In this way, the screening and monitoring frequency and extent can be automatically adjusted as the patient state changes. For conditional probability distribution initialization existing epidemiological data may be used [102].

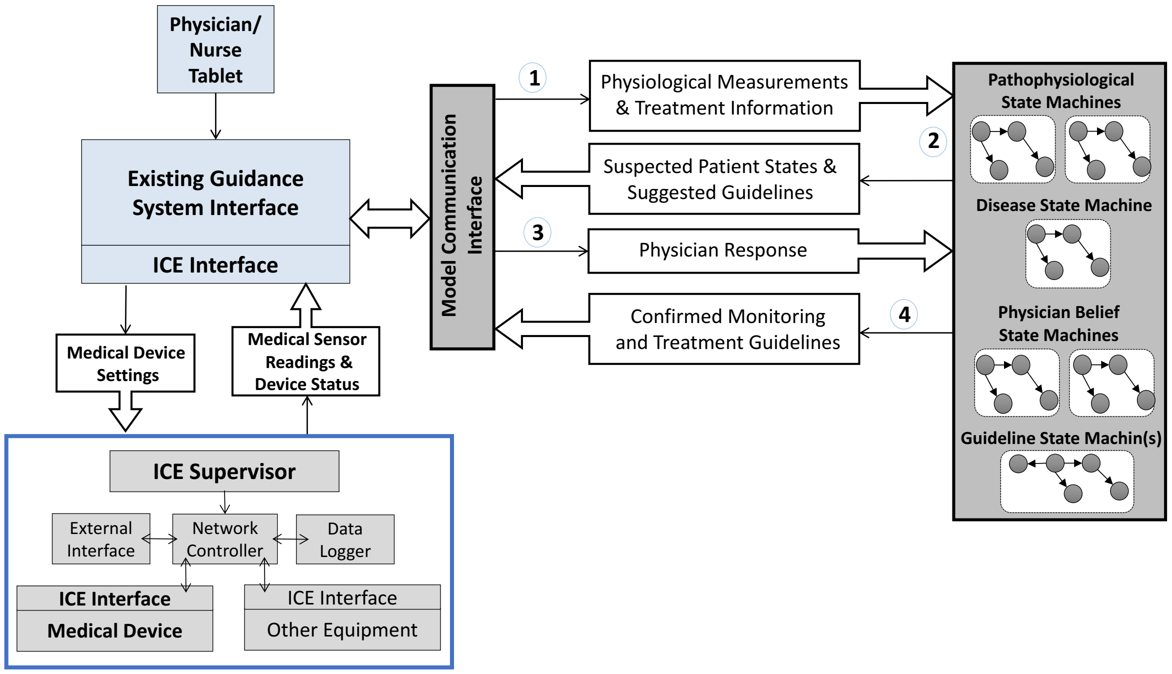
7.5 Medical System Safety

One of the potential benefits of our approach that can be exploited in the future is linking local medical device (ICE) system [50] to the developed pathophysiology-based models and have settings/-monitor values automatically communicated. Note that we may still require the human in the loop but integrating our executable models with ICE system allows automating getting and sending medical device interface data. As shown in Figure 7.1, the model-driven guidance system can interface with ICE supervisor. Note that the suspected patient state(s) and the corresponding guidelines are displayed for the attending physician; they can confirm the patient state(s) suggested by the system or choose a different diagnosis. If confirmed, the subset of corresponding best practice guidelines involving medical devices, can be translated to device settings and sent out to the ICE supervisor.

In this thesis, we assume that the medial best practice guidelines would produce the best clinical outcome and our work assist medical staff to avoid unintended deviations from these guidelines. However, as noted by Dr. Sandy Weininger,

> *Assuring the safety of automated medical systems requires addressing many level of governance and control, such as the practice of medicine, disease states and physiology, device states and responses. Linking all of these together (in some kind of mixed mode simulation) will be necessary in the future to assess the system performance as they are too complex to validate in vivo.*

Our modeling approach can be incorporated in such mixed-mode simulation, encoding and tracking the disease and pathophysiological state information according to best practice guidelines as well as physician belief at each state. These are indeed serious challenges that should be addressed in the future.



Physician/ Nurse Tablet
Existing Guidance System Interface
ICE Interface
Medical Device Settings
Medical Sensor Readings & Device Status
ICE Supervisor
External Interface
Network Controller
Data Logger
ICE Interface
Medical Device
ICE Interface
Other Equipment
Model Communication Interface
1
Physiological Measurements & Treatment Information
2
Suspected Patient States & Suggested Guidelines
3
Physician Response
4
Confirmed Monitoring and Treatment Guidelines
Pathophysiological State Machines
Disease State Machine
Physician Belief State Machines
Guideline State Machin(s)

Figure 7.1: Model Integration into Local Medical Device (”ICE”) System

7.6 User Acceptance

Medicine is complex and physician often needs to make decisions under subjective uncertainty. Trust can greatly affect the reliance on the guidance provided by our system. Trust in automation, or system-generated guidelines in our case, develops according to the information available [103]. *Providing information on model's confidence in its generated guidance* could foster acceptance/trust. As part of our future work, we will compute context-aware probability and risk values for different pathophysiological states encoded in the model, using state-based Bayesian analysis. Furthermore, we will identify the minimum required clinical data that must be available for each guideline item. The system can then update its confidence in the corresponding guideline item as additional clinical data becomes available, since new data may provide further evidence supporting the generated guidance.

Another way in which, trust can be further derived is from an understanding of the underlying mechanisms or observation of the system behavior [104]. We must provide an intuitive representation of not only the state of patient, but also the guidance process state (current and last *n* states). We can also display the process of generating the guideline items by revealing intermediate states/results in a way that can be easily understood by medical staff. In other words, we supplement the displayed guidance with *"What the Machine is Thinking?"*. Similar to flight control systems, we can also display the safe and effective next treatment actions and the corresponding parameters (***treatment safety envelope***). The chosen actions by medical staff can be overlaid on the safety envelope, for visual representation of potential deviations.

Bibliography

[1] Martin A Makary and Michael Daniel. Medical error the third leading cause of death in the us. *Bmj*, 353:i2139, 2016.

[2] Linda T. Kohn, Janet M. Corrigan, and Molla S. Donaldson. To err is human, building a safer health system, 1999.

[3] Orit Manor-Shulman, Joseph Beyene, Helena Frndova, and Christopher S Parshuram. Quantifying the volume of documented clinical information in critical illness. *Journal of critical care*, 23(2):245–250, 2008.

[4] https://www.ecri.org/press/Pages/ECRI-Institute-Announces-Top-10-Health-Technology-Hazards-for-2015.aspx.

[5] Mark A Musen, Blackford Middleton, and Robert A Greenes. Clinical decision-support systems. In *Biomedical informatics*, pages 643–674. Springer, 2014.

[6] Una Geary and Una Kennedy. Clinical decision-making in emergency medicine. *Emergencias*, 22:56–60, 2010.

[7] Haiyan Gao, Ann McDonnell, David A Harrison, Tracey Moore, Sheila Adam, Kathleen Daly, Lisa Esmonde, David R Goldhill, Gareth J Parry, Arash Rashidian, et al. Systematic review and evaluation of physiological track and trigger warning systems for identifying at-risk patients on the ward. *Intensive care medicine*, 33(4):667–679, 2007.

[8] http://statecharts.org/.

[9] Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.

[10] http://www.mdpnp.org.

[11] http://study.com/academy/lesson/selective-attention-definition-examples-quiz.html.

[12] Michael Imhoff and Silvia Kuhls. Alarm algorithms in critical care monitoring. *Anesthesia & Analgesia*, 102(5):1525–1537, 2006.

[13] Michael Imhoff, Silvia Kuhls, Ursula Gather, and Roland Fried. Smart alarms from medical devices in the or and icu. *Best Practice & Research Clinical Anaesthesiology*, 23(1):39–50, 2009.

[14] Andrew Y-Z Ou, Yu Jiang, Po-Liang Wu, Lui Sha, and Richard B Berlin. Preventable medical errors driven modeling of medical best practice guidance systems. *Journal of medical systems*, 41(1):9, 2017.

[15] Po-Liang Wu, Dhashrath Raguraman, Lui Sha, Richard B Berlin, and Julian M Goldman. A treatment validation protocol for cyber-physical-human medical systems. In *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, pages 183–190. IEEE, 2014.

[16] Po-Liang Wu, Lui Sha, Richard B Berlin, and Julian M Goldman. Safe workflow adaptation and validation protocol for medical cyber-physical systems. In *Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on*, pages 464–471. IEEE, 2015.

[17] Mor Peleg, Samson Tu, Jonathan Bury, Paolo Ciccarese, John Fox, Robert A Greenes, Richard Hall, Peter D Johnson, Neill Jones, Anand Kumar, et al. Comparing computer-interpretable guideline models: a case-study approach. *Journal of the American Medical Informatics Association*, 10(1):52–68, 2003.

[18] Paul A De Clercq, Johannes A Blom, Hendrikus HM Korsten, and Arie Hasman. Approaches for creating computer-interpretable guidelines that facilitate decision support. *Artificial intelligence in medicine*, 31(1):1–27, 2004.

[19] Ian N Purves. Prodigy: implementing clinical guidance using computers. *Br J Gen Pract*, 48(434):1552–1553, 1998.

[20] Samson W Tu, Mark A Musen, et al. Modeling data and knowledge in the eon guideline architecture. *Studies in health technology and informatics*, (1):280–284, 2001.

[21] Mor Peleg, Aziz A Boxwala, Omolola Ogunyemi, Qing Zeng, Samson Tu, Ronilda Lacson, Elmer Bernstam, Nachman Ash, Peter Mork, Lucila Ohno-Machado, et al. Glif3: the evolution of a guideline representation format. In *Proceedings of the AMIA Symposium*, page 645. American Medical Informatics Association, 2000.

[22] John Fox, Nicky Johns, Colin Lyons, Ali Rahmanzadeh, Richard Thomson, and Peter Wilson. Proforma: a general technology for clinical decision support systems. *Computer methods and programs in biomedicine*, 54(1):59–67, 1997.

[23] Silvia Miksch, Yuval Shahar, and Peter Johnson. Asbru: a task-specific, intention-based, and time-oriented language for representing skeletal plans. In *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, pages 9–19. Milton Keynes, UK, The Open University, Milton Keynes, UK, 1997.

[24] Hl7 rim: an incoherent standard. http://www.hl7.org/implement/standards/rim.cfm.

[25] Charles N MEAD Gunther SCHADOW. The hl7 reference information model under scrutiny. In *Ubiquity: technologies for better health in aging societies: proceedings of MIE2006*, volume 124, page 151. IOS Press, 2006.

[26] Donna L Hudson. Medical expert systems. *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[27] Shusaku Tsumoto and Hiroshi Tanaka. Automated discovery of medical expert system rules from clinical databases based on rough sets. In *KDD*, pages 63–69, 1996.

[28] Donna L Hudson and Maurice E Cohen. Fuzzy logic in medical expert systems. *Engineering in Medicine and Biology Magazine, IEEE*, 13(5):693–698, 1994.

[29] Steen Andreassen, Finn V Jensen, and Kristian G Olesen. Medical expert systems based on causal probabilistic networks. *International journal of bio-medical computing*, 28(1):1–30, 1991.

[30] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[31] George Hripcsak, Peter Ludemann, T Allan Pryor, Ove B Wigertz, and Paul D Clayton. Rationale for the arden syntax. *Computers and Biomedical Research*, 27(4):291–324, 1994.

[32] Peter Szolovits, Ramesh S Patil, and William B Schwartz. Artificial intelligence in medical diagnosis. *Annals of internal medicine*, 108(1):80–87, 1988.

[33] Susan Rea, Jyotishman Pathak, Guergana Savova, Thomas A Oniki, Les Westberg, Calvin E Beebe, Cui Tao, Craig G Parker, Peter J Haug, Stanley M Huff, et al. Building a robust, scalable and standards-driven infrastructure for secondary use of ehr data: the sharpn project. *Journal of biomedical informatics*, 45(4):763–771, 2012.

[34] Lisa Kelly, Bonnie Smith, Mark Davenport, and Bradley Scott. Presenting patient information by body system, December 10 2012. US Patent App. 13/709,478.

[35] Sholom M Weiss, Casimir A Kulikowski, Saul Amarel, and Aran Safir. A model-based method for computer-aided medical decision-making. *Artificial intelligence*, 11(1-2):145–172, 1978.

[36] Leon J Osterweil, George S Avrunin, Bin Chen, Lori A Clarke, Rachel Cobleigh, Elizabeth A Henneman, and Philip L Henneman. Engineering medical processes to improve their safety. In *Situational Method Engineering: Fundamentals and Experiences*, pages 267–282. Springer, 2007.

[37] Lori A Clarke, George S Avrunin, and Leon J Osterweil. Using software engineering technology to improve the quality of medical processes. In *Companion of the 30th international conference on Software engineering*, pages 889–898. ACM, 2008.

[38] Aaron G Cass, AS Lerner, Eric K McCall, Leon J Osterweil, Stanley M Sutton, and Alexander Wise. Little-jil/juliette: a process definition language and interpreter. In *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pages 754–757. IEEE, 2000.

[39] Stefan Christov, Bin Chen, George S Avrunin, Lori A Clarke, Leon J Osterweil, David Brown, Lucinda Cassells, Wilson Mertens, et al. Formally defining medical processes. *Methods of Information in Medicine*, 47(5):392, 2008.

[40] Bin Chen, George S Avrunin, Elizabeth A Henneman, Lori A Clarke, Leon J Osterweil, and Philip L Henneman. Analyzing medical processes. In *Proceedings of the 30th international conference on Software engineering*, pages 623–632. ACM, 2008.

[41] Gerard Holzmann. *Spin model checker, the: primer and reference manual*. Addison-Wesley Professional, 2003.

[42] George S Avrunin, Lori A Clarke, Leon J Osterweil, Stefan C Christov, Bin Chen, Elizabeth A Henneman, Philip L Henneman, Lucinda Cassells, and Wilson Mertens. Experience modeling and analyzing medical processes: Umass/baystate medical safety project overview. In *Proceedings of the 1st ACM international health informatics symposium*, pages 316–325. ACM, 2010.

[43] Donald J Reifer. Software failure modes and effects analysis. *IEEE Transactions on reliability*, 28(3):247–249, 1979.

[44] Thomas DeLong. Fault Tree Manual, 1970. Master's Thesis, Texas A&M University.

[45] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. Fault Tree Analysis, Methods, and Applications:A Review. *IEEE Transactions on Reliability*, R-34:194–203, 1985.

[46] Stefan C Christov, George S Avrunin, and Lori A Clarke. Considerations for online deviation detection in medical processes. In *Software Engineering in Health Care (SEHC), 2013 5th International Workshop on*, pages 50–56. IEEE, 2013.

[47] George S Avrunin, Lori A Clarke, Leon J Osterweil, Julian M Goldman, and Tracy Rausch. Smart checklists for human-intensive medical systems. In *Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on*, pages 1–6. IEEE, 2012.

[48] Stefan C Christov, Heather M Conboy, Nancy Famigletti, George S Avrunin, Lori A Clarke, and Leon J Osterweil. Smart checklists to improve healthcare outcomes. In *Proceedings of the International Workshop on Software Engineering in Healthcare Systems*, pages 54–57. ACM, 2016.

[49] http://www.mdpnp.org/uploads/Oct12_MD_PnP_White_Paper.pdf.

[50] ASTM F2761-09: Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) Part 1: General requirements and conceptual model, 2009.

[51] http://sourceforge.net/projects/mdpnp/.

[52] Insup Lee, George J. Pappas, Rance Cleavland, John Hatcliff, Bruce H. Krogh, Peter Lee, Harvey Rubin, and Lui Sha. High-Confidence Medical Device Software and Systems. *IEEE Computer*, 39:33–38, April 2006.

[53] John Hatcliff, Andrew King, Insup Lee, Alasdair MacDonald, Anura Fernando, Michael Robkin, Eugene Vasserman, Sandy Weininger, and Julian 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.

[54] Po-Liang Wu, Woochul Kang, Abdullah Al-Nayeem, Lui Sha, Richard B Berlin Jr, and Julian M Goldman. A low complexity coordination architecture for networked supervisory medical systems. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, pages 89–98. ACM, 2013.

[55] Andrew King, Sam Procter, Dan Andresen, John Hatcliff, Steve Warren, William Spees, Raoul Jetley, Paul Jones, and Sandy Weininger. An Open Test Bed for Medical Device Integration and Coordination. In *Proceedings of the 2009 International Conference on Software Engineering (ICSE)*, pages 141–151, 2009.

[56] Andrew King, Dave Arney, Insup Lee, Oleg Sokolsky, John Hatcliff, and Sam 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 Special Interest Group on Software Engineering, 2010.

[57] David Arney, Julian M. Goldman, Susan F. Whitehead, and Insup Lee. Synchronizing an X-ray and Anesthesia Machine Ventilator: A Medical Device Interoperability Case Study. In *Proceedings of International Conference on Biomedical Electronics and Devices*, pages 52–60, 2009.

[58] David Arney, Sebastian Fischmeister, Julian M. Goldman, Insup Lee, and Robert Trausmuth. Plug-and-Play for Medical Devices: Experiences from a Case Study. *Biomedical Instrumentation & Technology*, 43(4):313–317, 2009.

[59] David Arney, Miroslav Pajic, Julian M. Goldman, Insup Lee, Rahul Mangharam, and Oleg Sokolsky. Toward patient safety in closed-loop medical device systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '10, pages 139–148, New York, NY, USA, 2010. ACM.

[60] J-L Vincent, Rui Moreno, Jukka Takala, Sheila Willatts, Arnaldo De Mendonça, Hajo Bruining, CK Reinhart, PeterM Suter, and LG Thijs. The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure. *Intensive care medicine*, 22(7):707–710, 1996.

[61] Nancy Strzyzewski. Common errors made in resuscitation of respiratory and cardiac arrest. *Plastic Surgical Nursing*, 26(1):10–14, 2006.

[62] C. Guo, S. Ren, Y. Jiang, P. L. Wu, L. Sha, and R. B. Berlin. Transforming medical best practice guidelines to executable and verifiable statechart models. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10, April 2016.

[63] Maryam Rahmaniheris, PoLiang Wu, Lui Sha, and Richard R Berlin. An organ-centric best practice assist system for acute care. In *Computer-Based Medical Systems (CBMS), 2016 IEEE 29th International Symposium on*, pages 100–105. IEEE, 2016.

[64] http://www.heart.org/HEARTORG/.

[65] www.heart.org/acls.

[66] Arif Khwaja. Kdigo clinical practice guidelines for acute kidney injury. *The Nephron journals*, 120(4), 2012.

[67] https://uofi.box.com/s/45lsgunlfbw626avzgytk3d91cwz8w8y.

[68] https://uofi.box.com/s/r238jjypoyhnz03nveh87lxrekvhy9xw.

[69] http://carle.org.

[70] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[71] Mervyn Singer, Clifford S Deutschman, Christopher Warren Seymour, Manu Shankar-Hari, Djillali Annane, Michael Bauer, Rinaldo Bellomo, Gordon R Bernard, Jean-Daniel Chiche, Craig M Coopersmith, et al. The third international consensus definitions for sepsis and septic shock (sepsis-3). *Jama*, 315(8):801–810, 2016.

[72] Andrew Rhodes, Laura E Evans, Waleed Alhazzani, Mitchell M Levy, Massimo Antonelli, Ricard Ferrer, Anand Kumar, Jonathan E Sevransky, Charles L Sprung, Mark E Nunnally, et al. Surviving sepsis campaign: International guidelines for management of sepsis and septic shock: 2016. *Intensive care medicine*, 43(3):304–377, 2017.

[73] Yu Jiang, Lui Sha, Maryam Rahmaniheris, Binhua Wan, Mohammad Hosseini, Pengliu Tan, and Richard B Berlin Jr. Sepsis patient detection and monitor based on auto-bn. *Journal of medical systems*, 40(4):1–10, 2016.

[74] David Gallimore. Understanding the drugs used during cardiac arrest response. *Nurs Times*, 102(23):24–6, 2006.

[75] David S Wishart, Craig Knox, An Chi Guo, Dean Cheng, Savita Shrivastava, Dan Tzur, Bijaya Gautam, and Murtaza Hassanali. Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research*, 36(suppl 1):D901–D906, 2008.

[76] European Resuscitation Council et al. Part 6: advanced cardiovascular life support. section 6: pharmacology ii: agents to optimize cardiac output and blood pressure. european resuscitation council. *Resuscitation*, 46(1-3):155, 2000.

[77] MG Vermette and SF Perry. Effects of prolonged epinephrine infusion on blood respiratory and acid-base states in the rainbow trout: Alpha and beta effects. *Fish physiology and biochemistry*, 4(4):189–202, 1988.

[78] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, pages 87–124. Springer, 2004.

[79] Maryam Rahmaniheris, Lui Sha, Richard B. Berlin, and Julian M. Goldman. Towards a cyber-medical model for device configuration safety in acute care. In *Healthcare Innovation Conference (HIC), 2014 IEEE*, pages 118–124, Oct 2014.

[80] Maryam Rahmaniheris, PoLiang Wu, Lui Sha, Richard B. Berlin, and Julian M. Goldman. An organ-centric best practice assist system for acute care. *Submitted to Journal of Translational Engineering in Health and Medicine*, 2015.

[81] Maryam Rahmaniheris, Lui Sha, Richard B Berlin, and Julian M Goldman. Towards a cyber-medical model for device configuration safety in acute care. In *Healthcare Innovation Conference (HIC), 2014 IEEE*, pages 118–124. IEEE, 2014.

[82] Maryam Rahmaniheris, Woochul Kang, Lue-Jane Lee, Lui Sha, Richard B Berlin, and Julian M Goldman. Modeling and architecture design of an mdpnp acute care monitoring system. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*, pages 514–515. IEEE, 2013.

[83] Woochul Kang, PoLiang Wu, Maryam Rahmaniheris, Lui Sha, Richard B Berlin, and Julian M Goldman. Towards organ-centric compositional development of safe networked supervisory medical systems. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*, pages 143–148. IEEE, 2013.

[84] Richard Snodgrass. *The interface description language: definition and use*. Computer Science Press, Inc., 1989.

[85] International Electrotechnical Commission et al. International standard iec 601-1-2 medical electrical equipment part 1: General requirements for safety 2. *Collateral standard: Electromagnetic compatibility-requirements and tests*, 1993.

[86] Iso/ieee11073 point-of-care medical device communication standard. 1st ed., 2006. Available at www.ieee1073.org.

[87] Iso/ieee11073personal health devices standard (x73-phd). 1st ed., 2009. IEEE Standards Association webpage. Available at http://standards.ieee.org/.

[88] Bernhard Kaiser, Catharina Gramlich, and Marc Frster. State/Event Fault Trees- A Safety Analysis Model for Software-Controlled Systems. *Reliability Engineering & System Safety*, 92(11):1521 – 1537, 2007.

[89] http://mdpnp.org/MD_PnP_Program___Clinical_S.html.

[90] http://www.mdpnp.org/uploads/ASTM_F2761-09_ICE_Annex_B_Clinical_Scenarios.pdf.

[91] http://www.mdpnp.org/uploads/Capitol_Hill_NSF_CPS_MD_PnP_9July09.pdf.

[92] Ann S Lofsky. Turn your alarms on. *APSF Newsletter: The Official Journal of the Anesthesia Patient Safety Foundation*, 19(4):43, 2005.

[93] Cheolgi Kim, Mu Sun, Maryam Rahmaniheris, and Lui Sha. How to reliably integrate medical devices over wireless. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 85–87. IEEE, 2012.

[94] Crossing the quality chasm: A new health system for the 21st century, 2001.

[95] Sushrut S Waikar and Joseph V Bonventre. Creatinine kinetics and the definition of acute kidney injury. *Journal of the American Society of Nephrology*, 20(3):672–679, 2009.

[96] Anand Kumar, Daniel Roberts, Kenneth E Wood, Bruce Light, Joseph E Parrillo, Satendra Sharma, Robert Suppes, Daniel Feinstein, Sergio Zanotti, Leo Taiberg, et al. Duration of hypotension before initiation of effective antimicrobial therapy is the critical determinant of survival in human septic shock. *Critical care medicine*, 34(6):1589–1596, 2006.

[97] Finn V Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.

[98] Paul Dagum, Adam Galper, and Eric Horvitz. Dynamic network models for forecasting. In *Proceedings of the eighth international conference on uncertainty in artificial intelligence*, pages 41–48. Morgan Kaufmann Publishers Inc., 1992.

[99] Senthil K Nachimuthu and Peter J Haug. Early detection of sepsis in the emergency department using dynamic bayesian networks. In *AMIA Annual Symposium Proceedings*, volume 2012, page 653. American Medical Informatics Association, 2012.

[100] Eren Gultepe, Hien Nguyen, Timothy Albertson, and Ilias Tagkopoulos. A bayesian network for early diagnosis of sepsis patients: a basis for a clinical decision support system. In *Computational Advances in Bio and Medical Sciences (ICCABS), 2012 IEEE 2nd International Conference on*, pages 1–5. IEEE, 2012.

[101] Laura J Moore, Stephen L Jones, Laura A Kreiner, Bruce McKinley, Joseph F Sucher, S Rob Todd, Krista L Turner, Alicia Valdivia, and Frederick A Moore. Validation of a screening tool for the early identification of sepsis. *Journal of Trauma and Acute Care Surgery*, 66(6):1539–1547, 2009.

[102] Andrew Lever and Iain Mackenzie. Sepsis: definition, epidemiology, and diagnosis. *British Medical Journal*, 7625:879, 2007.

[103] John D Lee and Katrina A See. Trust in automation: Designing for appropriate reliance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 46(1):50–80, 2004.

[104] Kasey A Ackerman, Donald A Talleur, Ronald S Carbonari, Enric Xargay, Benjamin D Seefeldt, Alex Kirlik, Naira Hovakimyan, and Anna C Trujillo. Automation situation awareness display for a flight envelope protection system. *Journal of Guidance, Control, and Dynamics*, 2017.