View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Illinois Digital Environment for Access to Learning and Scholarship Repository

Towards a Systematic Software Architecture for Acute Care Support

Mu Sun, Maryam Rahmaniheris, Cheolgi Kim, Lui Sha
Dept. of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL
{*musun, rahmani1, cheolgi, lrs*}*@illinois.edu*

Richard B. Berlin Jr.
Dept. of Surgery,
College of Medicine and
Dept. of Computer Science
University of Illinois
Urbana, IL
rberlin@illinois.edu

Julian M. Goldman
Mass. General Hospital
and CIMIT
Cambridge, MA
jgoldman@mdpnp.org

***Abstract*—According to the Institute of Medicine (IOM) close to 100,000 safety-related medical incidents happens each year in US due to *preventable* medical errors [1]. These preventable medical errors are caused by lack of timely and comprehensive information about the patient that makes the proper and efficient application of best available knowledge difficult rather than lack of such knowledge. The preventable medical errors often occur due to slips and lapses in settings where the staff are overloaded and under stress. Our focus is on acute care scenarios where medical staff must make quick decisions based on the best available evidence. In this paper, we propose an acute care support system (MACMS) that aids the physician with monitoring and making decisions given the best available knowledge to decrease preventable medical errors. Such a system is possible due to the opportunity presented to us by Medical Device Plug-and-Play (MDPnP) to integrate information from different devices in an integrated clinical environment [2], [3]. In this paper, we also present a model driven approach for designing these acute care support systems by developing models that correspond closely to medical mental processes and a system architecture to support execution from these models.**

***Keywords*-medical systems; architecture; model driven design; model validation**

I. Introduction

Medical Device "Plug-and-Play" (MDPnP) interoperability opens the door to significantly mitigating the preventable medical errors [3]. According to the Institute of Medicine (IOM) at least 44,000 and perhaps as many as 98,000 Americans die in hospitals each year as a result of *preventable* medical errors [1]. These preventable medical errors are caused by improper application of best available medical knowledge rather than lack of such knowledge. According to "To Err is Human", a report published by IOM in 1999 [1], the preventable medical errors often occur due to slips and lapses in settings where the staff are overloaded and under stress such as emergency rooms and intensive care units. These slips and lapses include, but are not limited to error in diagnosis, error in administering a treatment and inadequate monitoring or followup of treatments.

One of the main contributing factors to slips and lapses in clinical environments is the lack of integrated information. According to a study from a Canadian group performed at 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 [4]. 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. Our goal is to facilitate and improve medical practice by reducing slips and lapses. We propose a monitoring system which exploit the benefits provided by safe MDPnP to combine and fuse information from different devices in an integrated clinical environment.

In this paper, we focus on the acute care environment where medical staff must make quick decisions based on the best available evidence. *We created an acute care support modeling language as well as an executable support system, the Medical Acute Care Monitoring System (MACMS), that aids the physician to monitor and making decisions given the best available knowledge to reduce medical errors.* MACMS tries to bring medical practice close to best practice by encoding well-accepted medical knowledge and medical community consensus. Furthermore, MACMS uses this encoded information to make suggestions for medical staff and notifies them of potential conflicts by:

Monitoring: using patient measurements (patient records and real-time patient physiological measures) to monitor correctness of a diagnosis and effectiveness of a treatment,

Suggesting: using patient measurements to suggest diagnosis and treatments according to best practice,

Adapting: changing the mode of operation of the system to be best suited for a current situation by selecting the appropriate set of patient physiologic signals to measure and using new measurements whenever they become available (medical device plug-and-play) to increase or decrease confidence in current diagnoses and treatments.

Integrated monitoring, suggesting, and adapting systems still have not made their way into safety-critical hospital environments. Medical information is considerably vast and complex. Even narrowing our focus to acute care settings with basic physiology (e.g. blood pressure, heart rate, etc.),

still calls for a systematic approach to collect and process the available information in a timely manner. Inter-operability of medical devices is critical for designing such monitoring system. MDPnP makes it possible for us to integrate information from different sources and coordinate medical actions by providing an integrated clinical environment (ICE) [2], [3]. There exist other safety-critical industries (e.g. chemical and material manufacturing and defense) that already have extensive experiences in designing monitoring systems [1]. It is common for safety-critical systems to process a large amount of information and data streaming from different types of sensors and act upon them [5], [6], [7]. However, there are unique features in medical systems that makes the traditional software architecture used to design monitoring systems inadequate:

Uncertainty Human physiology for acute care is founded on vital signs. This results in oversimplified and imprecise physiological models. Understanding the limitations of available models is critical for designing safe and efficient medical systems.

Procedure Flexibility Medical decisions are based on available information. The complexity and uncertainty of medicine makes definitive decisions difficult. Medicine must frequently err on the side of action and must make reasonable choices according to the best available knowledge. Furthermore, actions must be readily modified according to a patient's response.

Configuration Flexibility Medical systems need to inherently be run-time configurable. Different patients require different sensors and monitors. The set of medical devices including medical sensors required for a medical procedure can change throughout the procedure. Such dynamic configuration is not seen in other monitoring systems.

The MACMS software architecture addresses these challenges in medicine by using the following design principles. In order to deal with **uncertainty**, our system is reactive and is not predictive. The core of MACMS lies in continuous monitoring of vital signs through sensor signals because predicting patient physiology is difficult but detecting that a treatment is not working (based on expected physiological change) is simple. To allow **flexible procedures** and decisions, we let physicians override the system with reason. Furthermore, for all decisions of diagnosis and treatment made by the doctors, the system will monitor them for validity, consistency and effectiveness. Finally for **configuration flexibility**, as new medical sensors are plugged into the system and additional information becomes available, the system should check their consistency with the current information and use the new data to confirm or modify the current plan.

In the following sections, we will first describe the essential medical knowledge, core requirements, and our domain specific model (Section II). We then proceed to discuss the MACMS system architecture, how it executes the high level models, and how it interfaces with the physical world through real devices and input patient records (Section III). Afterwards, we provide a detailed case study, further illustrating all the pieces of MACMS operates with a small but complex acute care scenario (Section IV). Finally, we wrap up this paper by discussing related work (Section V) and conclusion (Section VI).

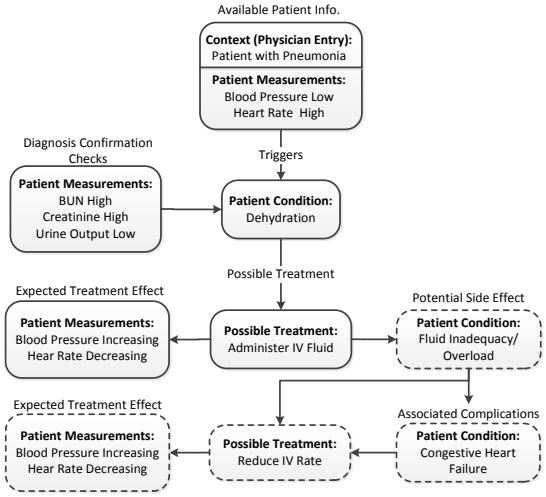

Figure 1. Simple Dehydration Example

II. Medical Domain Knowledge and System Specification

We now describe the necessary background for our architecture. We start by discussing some relevant points from the IOM and general medical community consensus. We will eventually describe some important principles of acute care practice and present our domain specific models that capture these principles.

A. Guiding Requirements of the IOM

Many of our major clinical requirements originate from an important subset of IOM recommendations [1], [8]. We highlight some of the most relevant and important ones.

1) *Avoid reliance on memory and vigilance (p.170 [1]).* Our system needs to be able to encode and retrieve clinical knowledge so less reliance is placed on human memory. According to a study from a Canadian group performed at 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 [4]. Fusion and processing of many partial information by the physician is an error-prone process due to the limitations of human memory and realtime processing capability. Our system should help support vigilance by automatically integrating the available

data, providing realtime processing of the information and continuous monitoring of patient.

2) *Improve access to accurate, timely information (p.174 [1])*. Our system needs to collect necessary information and react upon them in real time.
3) *Use simulations whenever possible (p.178 [1])*. Our system needs to be able to encode concrete case studies, so that we may study how well the system would perform with doctors by simulations.

B. Current Clinical Practice

Figure 1 shows a simple but common acute care scenario. A patient is admitted into intensive care. Basic vital sign measures are added, the blood pressure is low and the heart rate is high. This is most likely an indication of dehydration. Medical staff immediately proceed with this assumption, and IV saline fluid is given to the patient. While this scenario starts simply, the complexity immediately grows with possibilities. What if the patient wasn't actually dehydrated? What if additional measurements contradict the dehydration hypothesis? What if it is dehydration, but IV fluid does not improve the patient? What if it is not dehydration, but the IV fluid improves the patient anyway? At first glance through this limited case a clinical environment is very complex with many interactions between different decisions and many exceptions for every rule. Over the years, the medical community has developed an effective procedure to deal with this seemingly overwhelming complexity:

1) Always proceed with the best available information. Short term low-risk trial treatments may yield important information even if the trial does not succeed.
2) Always keep in the mind that current diagnoses may be wrong. Continue to validate diagnoses while proceeding with actions.
3) If it works, it is good. Patient condition improvement is the goal. The system assists the medical personnel by checking the physiological parameters of the patient with diagnosis for consistency.

This means that there may be intermediate reasoning steps to come up with diagnoses that lead to a treatment, but it is improvement that matters at the end. As long as we see that the patient is improving, one may proceed.

When a patient's condition worsens and without sufficient data to pinpoint the diagnosis, physicians often perform a trial treatment based on the available information and their experience and intuition. The top choice is often lower risk and reversible treatment that can be stopped, if patient's condition does not improve as expected. Close monitoring that integrates all the data in real time is particularly important during trials.

These principles provide a good model of the medical decision making. Figure 2 shows how the uncertainty decision process is reflected in the medical domain. Initially, the available set of patient information indicates an anomaly.

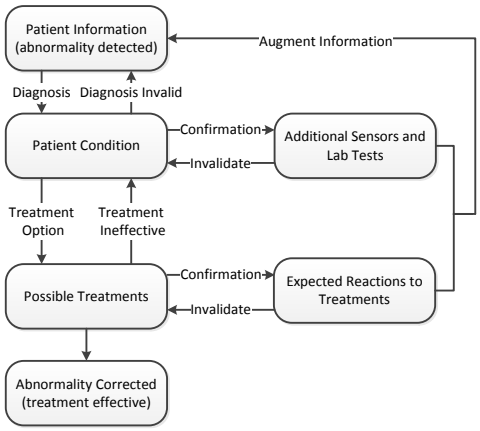

Figure 2. Iterative Medical Process for Dealing with Uncertainty

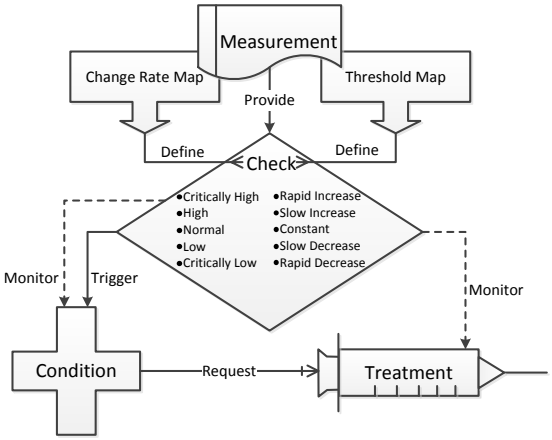

Figure 3. Encoding Medical Knowledge (Metamodel and Diagram Legend)

A likely diagnosis is made by the physician and a possible treatment is administered. In order to strengthen or invalidate a diagnosis, additional sensors and lab tests are ordered. These additional information actually augment the initial patient information allowing further diagnosis. In order to validate the effectiveness of treatment, the expected patient reactions and trends are monitored. In any uncertain stage of diagnosis or treatment, the physician is alerted as soon as physiological parameters of the patient show deterioration or failure to improve. In such case, the prior chain is reevaluated by the physician to consider other possibilities. If there was enough time, systematically, all known potentially effective treatments would have been tried until one is found that works. Essentially an exhaustive search.

C. Towards a Representative Model

We have just described a likely mental model used by many doctors. It is desirable to have a software architecture that operates and matches this closely to best aid doctors. To effectively use this mental model in software, we need to encode this medical knowledge. We have identified the main elements of the system to consist of (1) *available patient information (measurements)*, (2) *diagnosed patient conditions*, and (3) *provided treatments*. Figure 3 shows the metamodel based on the relations between these three main elements. We now dive into the individual model elements.

1) Patient Information: Patient information should describe all relevant information needed to make a diagnosis. Currently, we just consider patient information as the set of available measurements, which are numerical values. Examples of measurements include sensor readings (e.g. blood pressure, heart rate), lab values (e.g. creatinine, blood urea nitrate (BUN)), and other information (e.g. urine output, pain level). Normally, when diagnosing patient conditions, the exact measurements of blood pressure and heart rate are not required to make a diagnosis but just how much they deviate from normal. For each measurement, many times, the conditional checks on these measurements are based on ranges and normally do not require exact values. This means that for each measurement, there is a definition of critically high, high, low, critically low, which are abstractions of the ranges. A range map maps a range abstraction to a concrete measured value. For example, with heart rate, high may be mapped to 100 bpm, which defines the heart rate to be high if it exceeds 100 bpm. Similarly, we may not just need to reason about the thresholds of a measurement, but also about how fast it changes which is defined by a rate granularity (rapid increase, slow decrease, etc.).

Measurements are used by the system through Check blocks. These contain boolean clauses that can be used for various decisions in conditions and treatments. Normally, the Check statements are conjunctions of comparisons. A comparison could be a threshold comparison or a rate comparison. A comparison consists of a measurement from a plugged-in device and a range of either a threshold or a rate.

2) Diagnosed Patient Conditions: After we have modeled medical measurements and the statements that can be used to reason about them, we are ready to model patient conditions. Each Condition describes the initial measurements that trigger a diagnosis, the follow up monitoring logic that continues to confirm the diagnosis, and a possible set of treatments that can be used to treat this diagnosis. A condition is uniquely identified by its name. The trigger statement describes the checks that need to pass in order to have enough confidence to diagnose this patient condition. The monitor statement describes the continuous monitoring checks that need to be evaluated in order to validate a triggered condition diagnosis. Furthermore, each Condition lists the set of requested treatments which can potentially be administered in order to correct the patient condition.

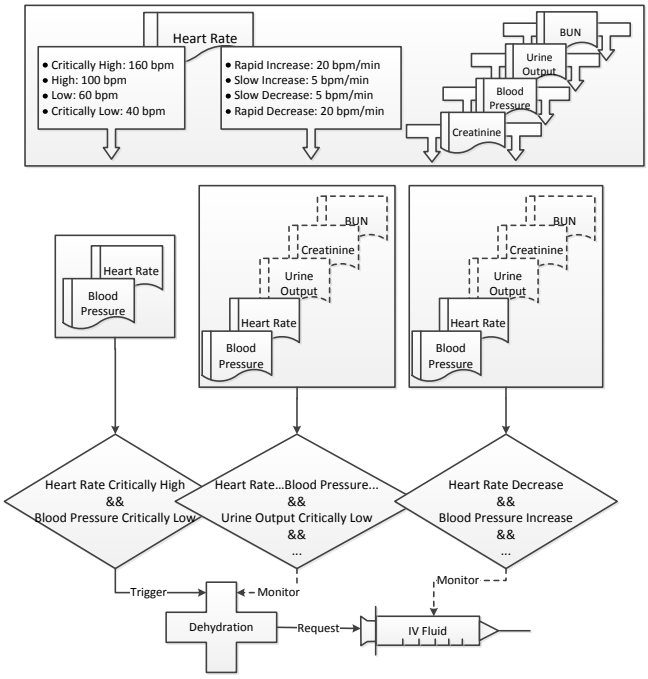

Figure 4. Model Instantiation (of metamodel form Fig. 3) for the Dehydration Example

3) Provided Treatments: Once a treatment is requested, it becomes decoupled of any conditions that requested it. The reasoning is that a diagnosis may change, but its requested treatments may continue as long as they improve patient vital signs. This means that a treatment has its own monitor that will continue to check vital sign trends to make sure the patient is responding as expected to the treatment. Furthermore, a treatment has a stop condition to suggest a potential stopping point of the treatment.

An example instantiation of our model for the dehydration scenario is shown in Figure 4. The measurements considered are heart rate and blood pressure. After it is diagnosed, additional measurements may be requested and used by the system once available such as BUN and urine output to confirm the diagnosis. The treatment of IV infusion also uses the heart rate and blood pressure to monitor its effectiveness.

D. Functional Requirements

For any system to use this model effectively, the following requirements summarizes the basic assumptions made on the model elements:

Measurement Updates Patient conditions should be checked when measurements exceed a set threshold

Measurement Additions The system should support the ability to incorporate new measurements from new plugged-in devices during run-time using the capabilities provided by MDPnP

Misdiagnosis / Change Diagnosis When the diagnosis changes, then these changes should be propagated to associated treatments so they can be properly modified after notifying medical personnel

Continuous Monitoring System should continue to monitor treatments and diagnoses as time elapses or as new measurements from new plugged-in devices or other sources (vital signs, lab results,...) become available, and report any contradictions to current prognosis

Prioritizing Conditions When conflicting conditions may be diagnosed there should be some method of deciding which one to proceed with first

Contraindicative Treatment All known treatment contraindications should be detected, notified, and stopped unless explicitly overwritten by medical personnel

III. Design Architecture

We have discussed the domain models, which are static descriptions of patient conditions and treatments and necessary logical checks on the measurements to monitor them over time. However, since the model describes when we can make diagnosis and how to monitor, the information contained in the model can be directly used to direct system execution for an acute care support system. We implemented the system in Java using Eclipse Modeling Framework (EMF) to manage the models.

A. Models as Execution Parameters

We used EMF to specify the information shown in the metamodel in Figure 3. EMF generates code that allows us to specify the acute care instance models (e.g. Figure 4) in an XMI format. Furthermore, EMF generates code that allows for the XMI model files to be deserialized into objects, so that we may directly access these object attributes to obtain model information during run-time.

Figure 5 shows how the deserialized objects of the model are used during system execution. Deserialized objects of the model are static, i.e. they only have attributes without any methods. Thus, to modularly define execution semantics using the information from these static objects, generic wrapper objects are defined. For example, in Figure 5, the description of dehydration (with trigger logic and monitoring logic) is deserialized into objects in the system, and a generic condition wrapper object manages all the subscriptions to the necessary measurements during execution. Also, the generic condition wrapper object has methods that are periodically invoked to check the trigger and monitoring conditions. Semantics of a *Check* is defined by a predicate that compares a current measurement with the corresponding threshold. For example, the statement `HeartRate HIGH` with `HIGH:100bpm`, will mean to check the latest heart rate measurement against 100 bpm. In summary, the key wrapper objects perform the following:

1) Measurement Wrappers:: For each type of measurement defined in a model instance, its measurement wrapper will subscribe to all the relevant devices (e.g. the heart rate measurement may subscribe to EKG Monitor, Pulse Oximeter, and Blood Pressure Cuff). Thus, a measurement wrapper has the ability to provide an aggregated measurement for this type. Furthermore, it will also contain the threshold definitions for high, low, rapid increase, decrease, etc. from the model, so it can also be directly queried for the whether a measurement is in a predefined range.

2) Condition Wrappers:: For each patient condition defined in a model instance, a condition wrapper will subscribe to all measurements required to evaluate trigger checks and monitor checks. It will also request appropriate treatments through the treatment manager (described later in subsection III-B). For each check statement, it will be evaluated by querying the appropriate measurement wrapper for whether a value is within a range. Condition wrappers also keeps track of when a condition becomes active (i.e. the trigger check evaluates to true).

3) Treatment Wrappers:: For each treatment option defined in a model instance, a treatment wrapper will subscribe to all required measurements, and perform the checks defined by the monitoring statement. It also keeps references to all conditions that requested to change the parameters of the treatment.

B. System Architecture

We have described the domain models and the wrappers that are used to give them appropriate semantics during execution. However, there still needs to be an overall architecture to manage these objects and their interactions, and ultimately, provide an interface to the real-world through physical devices. As we walk through the architecture, we will also discuss how the functional requirements described in Section II-D are met.

1) Managing Devices: Diagnosis and monitoring of treatment are all driven by medical measurements. Some measurements such as observations of the patient and lab tests may be input through a user interface or obtained through the electronic medical records (EMR). However, the most important real-time measurements are obtained through sensor devices. Normally only a small subset of critical sensors are connected to the patient in the beginning (e.g. oximeters, ECG, blood pressure monitor, etc.). But more sensors may be added depending on the current state of the patient (e.g. CVP sensor when heart failure is suspected). This means that we must support a plug-and-play system (Req. *Measurement Additions*). ICE standard provided by MDPnP group helps us reach this objective [2], [3]. In MACMS, we used Java front-end interfaces to medical devices. Most sensor devices

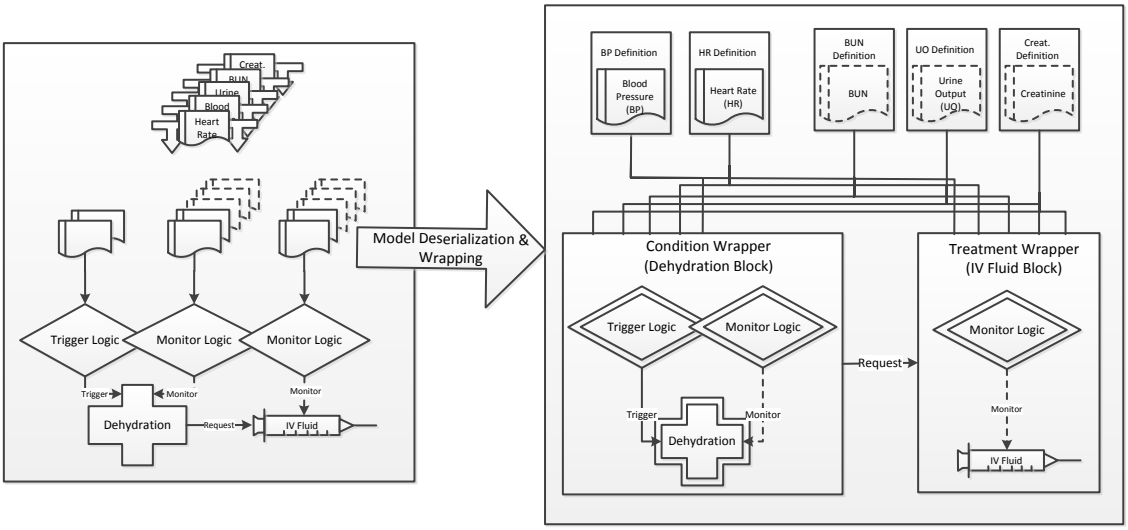

Figure 5. Using Models as Parameters for Execution

we use have standard external RS-232 interfaces, such as Pulse Oximeter and Infusion pump, and these can be easily interfaced to our system with standard serial communication.

Figure 6 shows the path from hardware sensors to the actual software measurements obtained in the system. The thing to note is that there is a many-to-many relation between sensors and measurements. For example, a pulse oximeter provides both heart rate and SpO_2 measures, and a blood pressure cuff provides both blood pressure and heart rate. Here, each devices provide two measurements, and heart rate is independently measured by two devices.

When a device plugs into the system, this will trigger an event to create a proxy object. Each hardware device store internally a model for its interface (i.e. the measurements it provides, and control parameters), and this interface model is sent to the system when the device is plugged in. For example, when an oximeter device plugs into the system, it will create a proxy oximeter object. The interface model will describe that the oximeter provides measurements for heart rate (HR) and SpO_2. The oximeter will then create separate measurement objects for the HR and SpO_2 that it provides. Each individual measured value such as HR will then register through the *Measurement Manager* to the corresponding measurement. This adds the HR measured value as a possible source of the HR measurement. The HR measurement manages the HR measured values from multiple sensors and combines them into one aggregate HR. For example, if two HR measured values comes independently from an oximeter and a blood pressure cuff, the HR Measurement interface will have either some prioritization or averaging algorithm to determine the reported HR.

2) Managing Conditions: The main component at the heart of our system are the diagnoses of patient conditions. The patient conditions provide the context of operations and monitoring. Figure 6 show the relations between the patient conditions and measurements. Each diagnosed patient condition is driven by a set of measurements. This means that each patient condition object references all of its dependent measurements. A new patient condition such as dehydration needs to subscribe to a set of required measurements for initial diagnosis: heart rate, blood pressure, etc., and the condition also needs to subscribe to a set of additional measurements for confirmation: urine output, BUN, etc. In the system once the dehydration object is created, it subscribes through the *Measurement Manager* to its list of required and additional measurements. The *Measurement Manager* then provides the dehydration object with the corresponding references to its requested measurements (Req. *Measurement Update*).

Each individual patient condition is kept track of by a dedicated *Condition Wrapper* object (described in Subsection III-A). Once the a Condition object such as dehydration has the references to its required measurements, the object's trigger logic will be rechecked whenever a measurement changes passed a threshold. When a measurement is updated, the object's trigger logic runs and checks all the trigger conditions (Req. *Measurement Updates*). For dehydration the trigger conditions are blood pressure low and heart rate high. This means that whenever the blood pressure drop and the heart rate increases, the dehydration trigger should be rechecked. Once a trigger condition passes, the Condition object becomes active.

The *Condition Manager* is responsible for keeping track

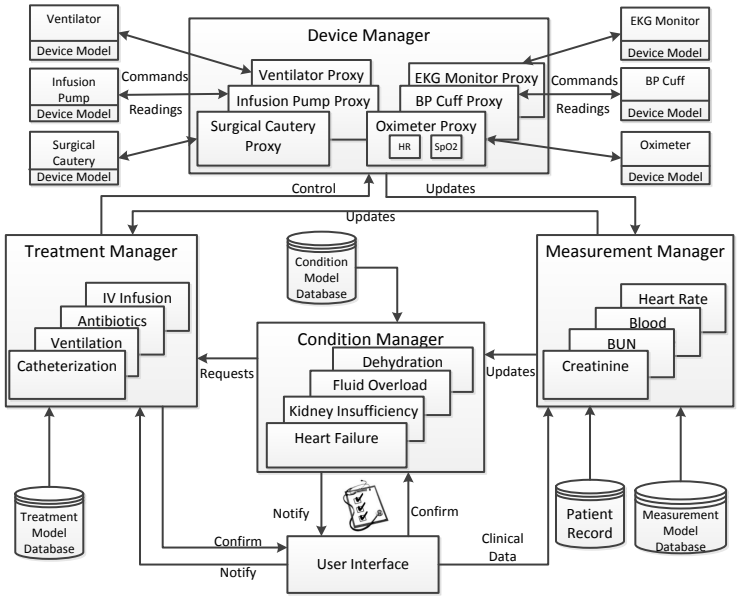

Figure 6. Overall MACMS Architecture

of all active Conditions. The *Condition Manger* is responsible for prioritizing active Conditions in terms of likelihood, criticality, and urgency (Req. *Prioritizing Conditions*). Furthermore, each active diagnosed Condition may indicate that other Conditions are likely to become active in the future, and the *Condition Manager* is also responsible for keeping track of potentially active Conditions in the future.

When a Condition object becomes active, it also sets up monitoring and requests treatment. Once a Condition sets up monitoring, it becomes time triggered and checks its vitals signs for inconsistencies on a periodic basis (Req. *Continuous Monitoring*). Monitoring for a newly activated condition may require additional sensors and additional lab tests. For dehydration, once the condition is activated, then it will request medical personnel to provide urine output, BUN samples, etc. The monitoring will use these measurements to ensure consistency of diagnosis when these sensors values change (requirement *Measurement Updates* in Section II). Aside from monitoring and checking, the conditions are also responsible for requesting the proper treatment through the *Treatment Manager*. If a continuous monitor detects an inconsistency when validating a condition based on measurements, it will notify medical staff of the potential inconsistency, and it will also notify associated conditions and dependent treatments (Req. *Misdiagnosis*).

3) Managing Treatment: Once a patient condition is diagnosed and confirmed by doctors it should also request appropriate treatments. As shown in Figure 6, the Condition sends a request to the *Treatment Manager* upon activation for the treatments to correct the condition. The *Condition Manager* keeps a list of references to the requested treatments for each condition. The TreatmentManager will grant the request if there is no contradiction between the requested treatment and the treatments already being administered (requirement *Contraindicative Treatment*). The TreatmentManager maintains the list of all the active treatments. The requested Treatment is triggered by the *Treatment Manager* and becomes active. The Treatment must subscribe to the Measurements through the *Measurements Manager* required to evaluate the patient progress in response to the treatment. The Treatment notifies the User Interface of any anomalies detected in the treatment.

IV. Discussion

In this section, we describe a detailed case study illustrating the potential complexities of the acute care process. The case study illustrates how a simple treatment for dehydration can slowly evolve into other conditions of fluid overload, heart failure, etc. We describe how the diagnosis and treatment knowledge in this medical process may be captured in our model, and furthermore, we describe how these models are used by MACMS during execution to adapt to the changing medical contexts.

A. Case Study Description

A 60 year old male, with history of mild heart disease is admitted to the emergency room with dehydration, related to pneumonia. Our system monitors patient vital signs continuously. When blood pressure and heart rate exceed the safe threshold the system should display dehydration as

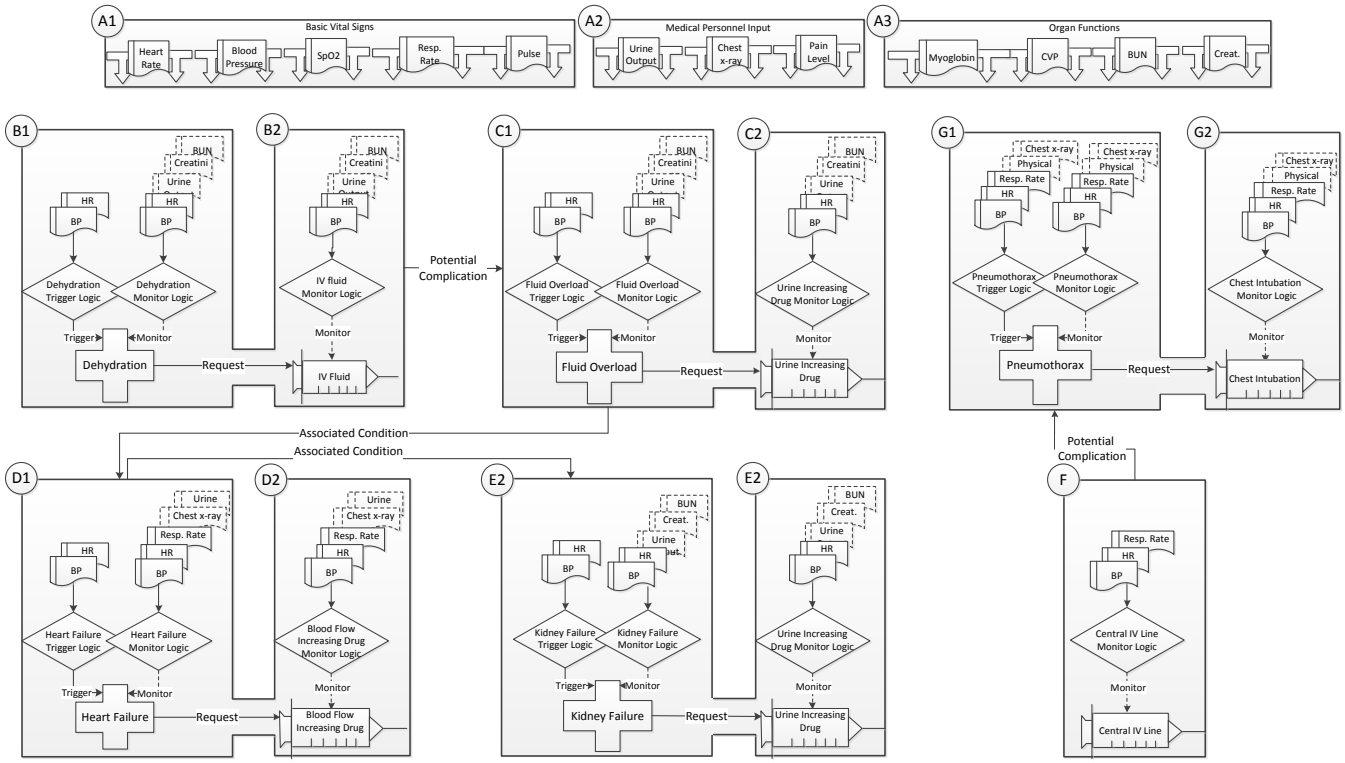

Figure 7. Use Case Scenario Model

one possible diagnosis for this patient who has a history of pneumonia. The system must confirm this diagnosis by requesting additional required measurements. In this scenario urine output and BUN measurements are required for confirmation. The patient is given intravenous fluid to correct the dehydration. However, due to a lapse of caution, he develops fluid overload. High pulse and low blood pressure indicates the condition which is detected by the system. The system should attend to the new condition by reducing the fluid rate and suggesting medication to increase urine output. After medical personnel modifies the treatment, a request for new urine output and BUN measurements should be issued by the system. When these measurements become available, the system should check their consistency with fluid overload diagnosis. The patient must be monitored for his physiological response to the new treatment. If the fluid overload leads to kidney insufficiency and congestive heart failure, the system should recognize their criticality and attend to them before continuing with the current diagnosis. In this scenario, there is no inconsistency between the treatments plans of the active condition due to the causality relation between them. In general, inconsistency in treatment plans should be detected and notified by the system.

Later, a central IV line is inserted to measure CVP (Central Venous Pressure) by the physician. Small pneumothorax (air in chest compressing lung) from insertion procedure occurs. The patient develops an acute cough, shortness of breath worsens, and has new chest pain, and little change in pulse and blood pressure suggests the condition. The system will request chest x-ray to confirm the diagnosis. The inspired oxygen level is increased by the medical staff but do not result in improvement of patient respiratory function. A chest tube is inserted by the physician for pneumothorax to correct respiratory function. However, chest tube becomes clogged and tension pneumothorax develops. Changes in both respiratory and pulse/blood pressure measures indicates the diagnosis which is displayed by the system. Medical staff and system work together to resolve the condition. Unfortunately, changes in IV rate and ventilator settings are not effective. Medical staff are alerted and directed by the system to unclog the tube. Tension pneumothorax is relieved after the tube is unclogged and patient physiological measurements return toward normal. Pneumonia remains primary problem, as pneumonia resolves, the ventilator feedback supplies less ventilator function, allowing patient to breathe more on own. Pneumonia resolves, patient is taken off the ventilator. The respiratory function and pulse/blood pressure measurement should be monitored afterward to ensure that the condition has resolved completely. It should be mentioned that all suggestions should be verified by the

medical staff before the system can move forward with the plan. Anytime the physician decides to overrides the system suggestion, the event should be recorded by the system for off-line analysis.

B. Modeling Elements of the Case Study

We introduced a metamodel for MACMS in section II to abstract the acute care process into elements necessary for a support system that can operate in a way that matches the mental model of a physician (see Figure 3). As mentioned before. The main elements of our model are *available patient information (measurements), diagnosed patient condition, and provided treatments*. This model is based on the interaction of these elements which is according to the medical process in acute care. In the scenario described above, patient information includes the information entered by the medical personnel, his existing chronic condition (Pneumonia), the vital signs including blood pressure and heart rate. This information will be augmented with additional measurements representing organ functions as they become available. In this scenario BUN and Creatinine test and urine output measure are requested by the system and their values are added to the patient information when they become available. The patient information model is divided into three main categories: vital signs (Figure 7-A1), medical staff entry (Figure 7-A2), and organ function measurements (Figure 7-A3).

Dehydration, Fluid Overload, Congestive Heart Failure and Kidney Failure models shown in Figure 7 are among the models required for our case scenario. The associated treatment models for each condition are also specified. IV Fluid, Urine Increasing and Blood Flow increasing medications are among the treatment models present in this scenario. The condition and treatment models specify the measurements required by the trigger logic and additional measurements required for monitor logic to confirm the condition and monitor the effect of treatment. For example according to Dehydration model (Figure 7-B1) BUN, Creatinine and urine output are the additional measurement needed for monitoring logic of this condition. As shown in Figure 7-B2, the IV fluid is the associated treatment for Dehydration condition. The same set of measurements are also used to monitor the treatment. Fluid Overload (Figure 7-C1) as one of the potential complications of IV Fluid treatment.

Congestive Heart Failure model (Figure 7-D1) is associated to Fluid Overload model (Figure 7-C1) and Kidney Failure model (Figure 7-E1) is associated to Heart Failure model (Figure 7-D1). This relations between models represent the causality relation between these conditions. Central IV Line model (Figure 7-F) describes the measurements that must be monitored when Cental IV Line is used in the scenario to measure CVP (Central Venus Pressure). Pneumothorax model (Figure 7-G1) is specified as a potential complication of the Cental IV Line model. As shown in Pneumothorax model blood pressure, heart rate, respiratory rate, chest x-ray and physical exam are required by Trigger and Monitor Logics of the condition. The Chest Intubation treatment model (Figure 7-G2) is associated with the Pneumothorax model. According to the Chest Intubation model, blood pressure, heart rate, respiratory rate, and physical exam is used by the treatment Monitor Logic.

The above set of models are selectively loaded into the system based on the current medical context. Furthermore, relations of potential complications of treatments and associated conditions are dynamically loaded into the system when the condition is diagnosed or the treatment is administered. The following subsection describes how the dynamic use of the models can be used to drive execution of an adaptive system.

C. Illustration of System Adaptation

When the system is initialized, the basic sensors such as blood pressure and pulse oximeter are plugged in. These provide the basic measurements of BP, HR, and SpO_2. Furthermore, because the patient has pneumonia, the dehydration condition object is loaded from the corresponding model (with the wrapper method described in Subsection III-B). This initial configuration is shown on the left of Figure 8(a).

Once dehydration is detected, the system will ask for doctors to confirm or ignore the diagnosis. If it is confirmed, then the system will also request that additional measurements and devices to be provided. The right of Figure 8(a) shows the system configuration after these doctor confirmed changes. An infusion pump is connected to the system to administer the saline fluid, and the corresponding treatment model is loaded with the necessary control parameters. Three additional measurements of BUN, Urine Output, and Creatinine are added. The treatment for IV fluid is added to provide support. A fluid overload condition (associated as a complication of giving IV fluid) is also loaded into the system as a potential complication to monitor.

Through what we just described, the system has gone through a mode change. Before, there was only one monitor for dehydration using three measurements, but after a diagnosis confirmation and addition of more devices and measurements, the system loop now operates on five measurements with two conditions being monitored and also providing treatment. This means that the system has essentially gone through a doctor-confirmed mode change that adapts to better matches and supports the current context of the operation.

The system mode change to diagnose dehydration and to administer IV treatment was completely anticipated, and the only uncertain part is the doctor's confirmation. However, there are also instances where the doctors may do things outside of the limited medical knowledge of the system. For example, in Figure 8(b), the initial system was setup with

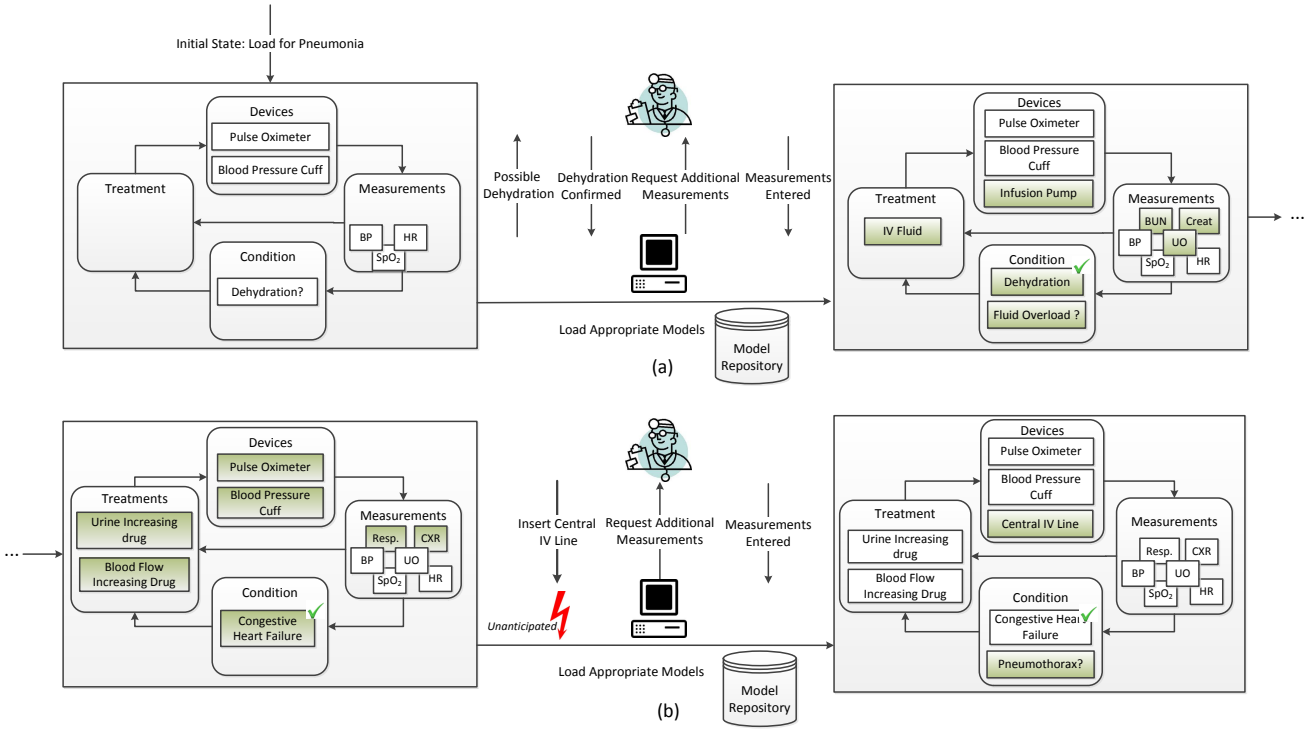

Figure 8. System Mode Adaptation Example with Reactions to Anticipated Events (a) and Unanticipated Events (b)

a continuous monitoring loop for congestive heart failure. The doctor decides to insert a central IV line to measure the central venous pressure (CVP). This is an unanticipated event since it is an asynchronous interruption from doctors. However, the system will similarly load in corresponding models for CVP and request additional measurements to check its safety. Furthermore, the likely complications of the central IV such as pneumothorax is also loaded into the system. In this example, the system was able to load in necessary elements to monitor the necessary measurements to best support a doctor's decision.

Of course, there is always the case when a doctor need to perform an action that has no support in the system. In this case, when things get too complex, the system can always fall back to a basic mode of operation. This basic mode will just monitor the most basic vital signs (heart rate, blood pressure, etc.) and provide notifications at signs of patient degradation and anomalies.

V. Related Work

The challenges of design of systems of interoperable medical devices are even harder to overcome compared to standalone devices. However, high confidence medical systems are necessary to avoid medical errors. The authors in [9] focus on the necessity of proper infrastructures on which networks of efficient and fully functioning interoperable plug-and-play medical devices can be built. According to [10] having a systems of interconnected plug-and-play medical devices in the ICU can provide the clinical staff with a more precise assessment of patient state. In [11] the authors also discuss the benefits of interoperable medical devices. The authors present the prototype of an interoperable medical system including a PCA pump and multiple monitoring devices to discuss the existing challenges and also the benefits of such systems. [12] and [13] present a framework to improve the flexibility and reconfigurability of prototype multi-device medical systems. In [14] the authors report their experience with a prototype plug-and-play synchronized medical system that includes the interconnection of an x-ray and anesthesia machine ventilator. In [15] the authors define the system safety properties in terms of patient state using an existing patient model.

Medical expert systems and clinical decision support systems assist medical staff with analysis of patient data and decision making process. These systems mostly use artificial intelligence, fuzzy logic and to deduction to make diagnoses and often do not go beyond making diagnosis. In some cases the system predict the future events based on the current diagnosis [16], [17], [18], [19] and [20]. In our system the main focus is monitoring the patient progress after an initial diagnosis is made by the physician. Fuzzy logic uses complex algorithms to essentially rank

diagnoses. Most of the time these are one shot and for chronic diagnoses. However, these ranking algorithms of fuzzy logic can be used later to improve the quality of prioritization schemes in our system [21], [22], [23]. There are many paper about modeling and verification for the medical domain [24], [25]. However, much of the focus has been on device level modeling, and not on capturing the physiological responses and monitoring for the patient. This builds upon the current modeling work for devices because this type of monitoring can be seamlessly integrated into existing systems to provide more robustness.

VI. Conclusion

We presented a software architecture for a support system for acute care, MACMS, that aims at reducing preventable medical errors specified in IOM reports. MACMS aids the physician to confirm diagnoses and monitor treatments given the available patient information. The system is based on the representation of medical process that closely matches mental model of physicians. The system collects and integrates all the measurements and patient data as they become available with simultaneous matching to the patient condition and expected physiological response of the patient. MACMS provides the medical personnel with realtime alert which are based on integrated patient data. We demonstrate the applicability of our model and execution framework using an acute care scenario.

The long term goals of our project is to provide a software architecture that is adaptive, dynamic, real-time, and provides the depth needed in a field of expanding criticality.

References

[1] L. T. Kohn, J. M. Corrigan, and M. S. Donaldson, "To err is human, building a safer health system," 1999.

[2] "ASTM F2761-09: Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) Part 1: General requirements and conceptual model," 2009. [Online]. Available: http://www.mdpnp.org/uploads/F2761_completed_committee_draft.pdf

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

[4] B. Pickering, V. Herasevich, A. Ahmed, and O. Gajic, "Novel representation of clinical information in the icu," pp. 116–131, April 2010. [Online]. Available: http://dx.doi.org/10.4338/ACI-2009-12-CR-0027

[5] C. M. Belcastro and S. R. Jacobson, "Future Integrated Systems Concept for Preventing Aircraft LOss-of-Control Accidents," *AIAA Guidance Navigation and Control Conference*, no. August, pp. 1–16, 2010. [Online]. Available: http://hdl.handle.net/2060/20100031283

[6] C. M. Belcastro and J. V. Foster, "Aircraft Loss-of-Contol Accident," *Control*, no. August, pp. 1–39, 2010. [Online]. Available: http://ntrs.nasa.gov/details.jsp?R=526827

[7] C. M. Belcastro, "Validation and Verification of Future Integrated Safety Critical Systems Operating Under Off-Nominal Conditions," *AIAA Guidance Navigation and Control Conference*, 2010.

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

[9] I. Lee, G. J. Pappas, R. Cleavland, J. Hatcliff, B. H. Krogh, P. Lee, H. Rubin, and L. Sha, "High-Confidence Medical Device Software and Systems," *IEEE Computer*, vol. 39, pp. 33–38, April 2006.

[10] K. Grifantini, "Plug-and-Play Hospitals: Medical devices that exchange data could make hospitals safer," *MIT Technology Review*, July 2008.

[11] D. Arney, S. Fischmeister, J. M. Goldman, I. Lee, and R. Trausmuth, "Plug-and-Play for Medical Devices: Experiences from a Case Study," *Biomedical Instrumentation & Technology*, vol. 43, no. 4, pp. 313–317, 2009.

[12] A. King, D. Arney, I. Lee, O. Sokolsky, J. Hatcliff, and S. Procter, "Prototyping Closed Loop Physiologic Control with the Medical Device Coordination Framework," in *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care*. ACM Special Interest Group on Software Engineering, 2010, pp. 1–11.

[13] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, and S. Weininger, "An Open Test Bed for Medical Device Integration and Coordination," in *Proceedings of the 2009 International Conference on Software Engineering (ICSE)*, 2009, pp. 141–151.

[14] D. Arney, J. M. Goldman, S. F. Whitehead, and I. Lee, "Synchronizing an X-ray and Anesthesia Machine Ventilator: A Medical Device Interoperability Case Study," in *Proceedings of the 2009 International Conference on Biomedical Electronics and Devices (BioDevices)*, 2009, pp. 52–60.

[15] D. Arney, M. Pajic, J. M. Goldman, I. Lee, R. Mangharam, and O. Sokolsky, "Toward patient safety in closed-loop medical device systems," in *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, ser. ICCPS '10. New York, NY, USA: ACM, 2010, pp. 139–148.

[16] P. Durieux, R. Nizard, P. Ravaud, N. Mounier, and E. Lepage, "A Clinical Decision Support System for Prevention of Venous Thromboembolism," *JAMA: The Journal of the American Medical Association*, vol. 283, no. 21, pp. 2816–2821, 2000.

[17] A. A. Montgomery, T. Fahey, T. J. Peters, C. MacIntosh, and D. J. Sharp, "Evaluation of computer based clinical decision support system and risk chart for management of hypertension in primary care: randomised controlled trial," *BMJ*, vol. 320, no. 7236, pp. 686–690, 2000.

[18] D. L. Hunt, R. B. Haynes, S. E. Hanna, and K. Smith, "Effects of Computer-Based Clinical Decision Support Systems on Physician Performance and Patient Outcomes," *JAMA: The Journal of the American Medical Association*, vol. 280, no. 15, pp. 1339–1346, 1998.

[19] K. Kawamoto, C. A. Houlihan, E. A. Balas, and D. F. Lobach, "Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success," *BMJ*, vol. 330, no. 7494, p. 765, 2005.

[20] D. W. Bates, G. J. Kuperman, S. Wang, T. Gandhi, A. Kittler, L. Volk, C. Spurr, R. Khorasani, M. Tanasijevic, and B. Middleton, "Ten commandments for effective clinical decision support: making the practice of evidence-based medicine a reality," *Journal of the American Medical Informatics Association*, vol. 10, no. 6, pp. 523 – 530, 2003.

[21] J. Warren, G. Beliakov, and B. van der Zwaag, "Fuzzy logic in clinical practice decision support systems," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000.

[22] R. John and P. Innocent, "Modeling uncertainty in clinical diagnosis using fuzzy logic," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 6, 2005.

[23] G. Beliakov and J. Warren, "Appropriate choice of aggregation operators in fuzzy decision support systems," *IEEE Transactions on Fuzzy Systems*, 2001.

[24] I. Lee, G. Pappas, R. Cleaveland, J. Hatcliff, B. Krogh, P. Lee, H. Rubin, and L. Sha, "High-confidence medical device software and systems," *Computer*, vol. 39, no. 4, pp. 33 – 38, april 2006.

[25] R. P. Jetley, C. Carlos, and S. P. Iyer, "A case study on applying formal methods to medical devices: computer-aided resuscitation algorithm," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 5, pp. 320–330, 2004.