Specifying and Analyzing Workflows for Automated Identification and Data Capture

Elsa L. Gunter, Ayesha Yasmeen, Carl A. Gunter, and Anh Nguyen
University of Illinois
{egunter,yasmeen,cgunter,anguyen7}@illinois.edu

Abstract

Humans use computers to carry out tasks that neither is able to do easily alone: humans provide eyes, hands, and judgment while computers provide computation, networking, and storage. This symbiosis is especially evident in workflows where humans identify objects using bar codes or RFID tags and capture data about them for the computer. This Automated Identification and Data Capture (AIDC) *is increasingly important in areas such as inventory systems and health care. Humans involved in AIDC follow simple rules and rely on the computer to catch mistakes; in complex situations this reliance can lead to mismatches between human workflows and system programming. In this paper we explore the design, implementation and formal modeling of AIDC for vital signs measurements in hospitals. To this end we describe the design of a wireless mobile* medical mediator *device that mediates between identifications, measurements, and updates of Electronic Health Records (EHRs). We implement this as a system* Med2 *that uses PDAs equipped with Bluetooth, WiFi, and RFID wireless capabilities. Using Communicating Sequential Processes (CSP) we jointly specify workflow and computer system operations and provide a formal analysis of the protections the system provides for user errors.*

1. Introduction

Automated Identification and Data Capture (AIDC) provides support for a growing collection of tasks in which sensor systems, often manipulated by humans, are used to identify objects and collect measurements about them. Identification uses tools like bar codes or Radio Frequency Identification (RFID) tags with optical or radio readers. Data capture involves both manual input and an increasingly rich collection of measurement devices often linked wirelessly to computer networks. While workflows like the checkout lanes at grocery stores are fairly simple and have become well-established, in other areas AIDC is only beginning to see practical application because reliability, flexibility, and cost savings have not yet been adequately addressed. A particular case in point is medical workflows where the identification of readings and samples is of safety critical importance and workflows are quite complex. Better analysis techniques are needed to help design workflows and systems so that these concerns can be addressed with high confidence.

In this paper we explore modeling techniques based on process algebras to better design and analyze human workflows and the computer and networking systems that support them. Our primary focus and contribution is on the analysis of the gap between the human workflow instructions, which must be comparatively simple, and the *protection envelope* comprising the collection of erroneous behaviors the computer is able to detect. It is typically impossible to detect all types of human error using storage and sensor data, so it is essential to identify exactly which types of conditions lie outside of the protection envelope and address these through system extensions or user training. As systems become more complex and safety critical, it is increasingly undesirable to base the detection of errors on operational experience and hence formal means of analysis provide a premium.

As a simple example, consider the move from current grocery store checkout lines which use attendants to scan items to ones that encourage the customer to perform this task. A typical system of this type might, for instance, provide a scale onto which the user places items after scanning them; if there are inconsistences between the expected weight of the item scanned and the measured weight of the item placed on the scale, then an attendant may be called to review the situation. This example illustrates the combination of automated identification (the scan), data capture (the weighing), and the envelope of protection (exceptional conditions arising from inconsistency between identification and data

capture). The default behavior is simple, scan items and put them on the scale, but a broader protection envelope is supported if this workflow is violated. The situation is made more complex when additional issues arise, such as a customer using their own cloth bag (to avoid choosing paper or plastic): can (or should) the system be designed to account for the added weight of the customer's bag on the scale without generating a call-attendant signal?

This paper introduces a methodology based on Communicating Sequential Processes (CSP) that provides basic vocabulary for AIDC. We show that this language can be effective in characterizing and analyzing the human workflow, the system protection envelope, and the implementation for complex and potentially valuable AIDC systems. We demonstrate this with a device we call a "medical mediator" which can be carried by a nurse or other clinical personnel and used to identify devices and subjects (patients) so their data can be automatically recorded into the hospital information system with a well-understood set of rules about the protection envelope of the device and workflow.

The paper is divided into seven sections. We begin in Section 2 with background material on CSP, how it can be used to model workflow, and the opportunity and challenge of AIDC in healthcare. Section 3 describes the design of the medical mediator and Section 4 sketches a prototype implementation of a medical mediator using PDAs with WiFi, Bluetooth, and RFID. In Section 5 we describe our modeling technique for AIDC using CSP and apply it to the medical mediator by specifying human workflow and the protection envelope. Section 6 sketches safety properties of the medical mediator derived from our three specifications. Section 7 describes some of the related work on workflow modeling and AIDC and Section 8 provides some conclusions.

2. Background

Since our focus will be on using CSP to model AIDC in health care, we need some background on CSP notations, how CSP can be used to model workflows, and the opportunity and challenge of AIDC in health care.

Communicating Sequential Processes (CSP). CSP is an algebraic notation for concurrency and communication that is used to describe processes in terms of their patterns of behavior. We will try to make our discussion self-contained but more details and examples can be found in [11, 21].

A simplified grammar of CSP is given below. It is based on a primitive set of events e and booleans b.

$$P, Q ::= e \rightarrow P \mid \mathsf{Stop} \mid \mathsf{Skip} \mid \mathsf{if}\ b\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid P \setminus A \mid P \triangle Q \mid P;Q \mid P \,|||\, Q \mid P;Q \mid P[\![A]\!]Q \mid P \,\Box\, Q \mid P \sqcap Q$$

The process $e \rightarrow P$ first does e and then behaves like P. The constant Skip indicates successful completion while Stop indicates a deadlocked process. The process if b then P else Q behaves like P if b holds and like Q otherwise. The process $P \setminus A$ behaves like P but with all elements of A removed from its traces as if they have been internalized. The process $P \triangle Q$ behaves like P but becomes Q on interrupt. The process $P;Q$ behaves like P until P terminates, and then behaves like Q. The process $P \,|||\, Q$ denotes the interleaved parallel composition of the two processes P and Q. In the process $P|[A]|Q$, the processes P and Q must synchronize on the events in the event alphabet A but execute with parallel interleaving otherwise. The processes $P \,\Box\, Q$ and $P \sqcap Q$ each behave either like P or Q, but the choice between them is made by the environment for $P \,\Box\, Q$ and internally for $P \sqcap Q$. The operators $|||$, $\Box$, and $\sqcap$ are associative and commutative. We will abbreviate $P(x_1) \,\Box\, \ldots \,\Box\, P(x_n)$ by $\underset{x \in X}{\Box} P(x)$ where $X = \{x_1, \ldots, x_n\}$, and similarly for $|||$ and $\sqcap$. (All sets will be assumed to be finite.) We will further abbreviate $\underset{\{C.x \mid x \in X\}}{\Box} C.x \rightarrow P(x)$, where X is an understood set of values for a communication, by $C?x \rightarrow P(x)$ meaning the possible receipt of a value, and write $C!v \rightarrow P$ for $C.v \rightarrow P$, representing the sending of a value v. We will refer to the modifier C in a set of events $\{C.x | x \in X\}$ as a communication channel, or channel for short.

Using CSP to Model Workflow. Perhaps the easiest way to understand CSP and its potential for workflow modeling is to work through an example. We do this for an identification workflow used by the U.S. Transportation Safety Agency (TSA) [2] and illustrate a (known) procedural gap using CSP notation. Our goal here is simply to show how concurrency contructs can be used to express sometimes subtle aspects of human workflow using an example that will be familiar to most readers. The bulk of the paper will focus on more complex and specialized workflows in health care.

TSA provides screening of passengers and their carry-on baggage in a checkpoint that passengers must traverse to reach boarding gates. At the checkpoint, the passenger is required to place their bags onto a conveyor belt, where they are screened by X-ray, while the passenger steps through a metal (and possibly chemical) detector. After stepping through the metal detector, a passenger may be selected by TSA for further screening. In this case the passenger is asked to identify their

bags and step to a separate screening zone. A security gap arises in this workflow because the workflow does not bind the passenger to his baggage before the screening step is executed, thus allowing the passenger to potentially select a different bag than the one they were originally carrying. The check against this happening is the assumption that the selected passenger's bag would remain unclaimed while the other passenger whose bag was taken would complain, thereby alerting the the TSA to a problem. However, if the person selected has an accomplice, they could enter together and swap bags if one was pulled for further screening. The end association of bag with passenger has failed to match corresponding identities; after the bag selection is complete, the TSA agent does not know the identity of the bag selected relative to the passenger selected. It is arguable that there is an error in the workflow specification: the workflow should specify that the TSA agent makes the choice of the bag for the passenger to reclaim, and not the passenger. Let us see how the workflow in particular, and this issue in general, can be modeled formally using CSP.

We model the specification of the passenger and TSA agent as follows. We assume events for clearing a passenger Clr, selecting a passenger Sel, and getting a bag Bag, and processes for going to the gates Gate, or for going to the waiting area Wait (for further screening).

$$\begin{aligned}\mathsf{Pass}(p) = {} & (\mathsf{Clr}.p \longrightarrow \sqcap_{b \in Bags} \mathsf{Bag}.b \longrightarrow \mathsf{Gate}(p,b)) \\ & \Box(\mathsf{Sel}.p \longrightarrow \sqcap_{b \in Bags} \mathsf{Bag}.b \longrightarrow \mathsf{Wait}(p,b)) \\ \mathsf{TSA} = {} & \Box_{p \in Pass} \ (\mathsf{Clr}.p \longrightarrow \Box_{b \in Bags} \mathsf{Bag}.b \longrightarrow \mathsf{TSA}) \\ & \sqcap(\mathsf{Sel}.p \longrightarrow \Box_{b \in Bags} \mathsf{Bag}.b \longrightarrow \mathsf{Screen}(p,b))\end{aligned}$$

This allows for the following implementation (*failure refinement* in CSP terminology) in which a suspicious passenger p plans ahead to swap bags with accomplice passenger a.

$$\begin{aligned}&(\mathsf{Clr}.p \longrightarrow \mathsf{Bag}.b_p \longrightarrow \mathsf{Gate}(p,b_p)) \\ &\Box(\mathsf{Sel}.p \longrightarrow \mathsf{Bag}.b_a \longrightarrow \mathsf{Wait}(p,b_a))\end{aligned}$$

One strategy to address this problem is to change the specification of the workflow for the passenger to:

$$\begin{aligned}\mathsf{Pass}(p) = {} & \sqcap_{b \in Bags} \ \mathsf{Bag}.b \longrightarrow (\mathsf{Clr}.p \longrightarrow \mathsf{Gate}(p,b)) \\ & \Box(\mathsf{Sel}.p \longrightarrow \mathsf{Wait}(p,b))\end{aligned}$$

Practically speaking, this means not selecting the passengers for extra screening until after they have selected the bags they plan to take to the gate with them. The specification of the TSA would now need to be changed to reflect the order of bag selection and passenger screening selection. By changing the specification this way, we are (more) assured that the bag has been identified as the one that the passenger plans to take with them. However, this still does not guarantee that the bag is the one that the passenger brought with them. An alternative strategy is to change specification of the TSA's workflow:

$$\begin{aligned}\mathsf{TSA} = {} & \Box_{p \in Pass} \ (\mathsf{Clr}.p \longrightarrow \Box_{b \in Bags} \mathsf{Bag}.b \longrightarrow \mathsf{TSA}) \\ & \sqcap(\mathsf{Sel}.p \longrightarrow \mathsf{Bag}.b_p \longrightarrow \mathsf{Screen}(p,b_p))\end{aligned}$$

This specifies that the TSA officer is required to select the bag of the passenger. Of course, a proper execution requires that the TSA agent remember or somehow record which bag the passenger had when he reached the checkpoint so the selection can be made accurately if it needs to be made.

AIDC in Health Care. Identification is especially important in health care. Everyone has heard horror stories about patients who got an operation intended for another patient [6], died because they got the wrong medication [14], or had surgery that mis-identifies a limb that needed to be amputated. Samples must be very carefully handled in labs and there is an especially important gap between the collection of a reading and its entry into an Electronic Health Record (EHR). As a result, health care facilities such as hospitals, labs, and clinics have rigorous workflows for clinical personnel that involve careful identification. Clinicians, patients and samples are all labeled with badges, arm tags and stickers respectively for use in many procedures.

There are efforts to improve efficiency in hospitals by using AIDC and we will overview a few in our related work section. Our primary focus in this paper concerns using tags to facilitate EHR updates from readings a clinician takes from a patient using a medical device. Wireless communication capabilities such as WiFi, Blue-tooth, and RFID are changing hospital workflows. To see the trend, first note that much current workflow is comparatively manual. For instance, a nurse collects the weight of a patient on a mechanical scale and writes it on into a paper file. With the emergence of EHRs, the nurse then enters this information into an EHR at a later stage. But if wireless medical devices were used, the workflow could have been streamlined. For instance, a wireless scale has a Blue-tooth link to a computer and the reading is transfered directly to the computer with the nurse providing other necessary identifications. *Or* the scale connects by Blue-tooth to a Personal Digital Assistant (PDA) carried by the nurse, displays the reading and enters it via a WiFi link into the EHR. *Or*, in the case of our system in this paper, a scale / pulse oximeter connects by Blue-tooth to a PDA (which we will subsequently refer to as a *medical mediator*), which uses an RFID reader to identify the patient and the clinician before entering the results into an EHR via a WiFi link. We add the additional step of identifying the device making

the reading to obtain the following:

Nurse Workflow: Identify the patient by scanning the patient's identification tag, then identify the device to be used to take a reading by scanning its identification tag. View the information to verify whether the correct entities were identified. Take a reading and check the reading to see if it is acceptable for entry into the EHR and approve if it is.

This is the type of workflow a human can understand and follow; it can be augmented by some simple recovery instructions like using a reset function if the entities are not correctly identified or the reading is not satisfactory. It can be expressed in CSP without great complexity as follows:

$$
\begin{aligned}
&\mathsf{Nurse}(n,m) = \\
&\quad \mathsf{HCI}^n_m!\mathsf{GetID} \rightarrow \sqcap_p(\mathsf{HCI}^n_m!(\mathsf{RFIDChan}^{n,m}_p) \rightarrow \\
&\quad \sqcap_d(\mathsf{HCI}^n_m!(\mathsf{RFIDChan}^{n,m}_d) \rightarrow \mathsf{HCI}^n_m?x \rightarrow \\
&\quad\ \mathsf{if}\ x = (\mathsf{Name}(p), \mathsf{Name}(d)) \\
&\quad\ \mathsf{then}\ (\mathsf{HCI}^n_m!\mathsf{Yes} \rightarrow \mathsf{TakeCkReading}(n,m,p,d)) \\
&\quad\ \mathsf{else}\ \ \mathsf{if}\ \ x = \mathsf{Error}\ \ \mathsf{then}\ (\mathsf{HCI}^n_m!\mathsf{OK} \rightarrow \mathsf{Skip}) \\
&\qquad\ \mathsf{else}\ (\mathsf{HCI}^n_m!\mathsf{No} \rightarrow \mathsf{Skip})))
\end{aligned}
$$

where HCI^n_m is the human-computer interface for the PDA m used by the clinician n and $\mathsf{RFIDChan}^{n,m}_x$ is the connection between the RFID tag on x (person p or device d) and the medical mediator m provided by the clinician n bringing the RFID reader in range of the RFID tag. The process TakeCkReading for collecting the reading from the device is described in Section 5.

Our primary interest is in what happens when we step outside of the simple human workflow and consider the many exceptional conditions that plague practical operational deployment and implementation. For example, what happens if the nurse scans the patient's ID more than once, or skips the identification of the device, or does the identifications in a different order, or identifies the wrong type of object (like herself as patient), and so on. In these cases one hopes that the computer can correct many errors, but it is generally impossible to correct all of them (for example a nurse who takes a reading of a patient different from the one identified) so it is important to know exactly which ones will *not* be caught.

3. Medical Mediator

As mentioned earlier, it is not unusual for a nurse to take a measurement from a patient, record the reading into a piece of paper, and enter it into patient's EHR at a later stage. This workflow is not only inefficient but it also creates many opportunities for mistakes. In the meantime there has been excellent progress on developing wireless medical devices that automatically transfer their readings to a nearby station. Research-oriented efforts include Harvard CodeBlue [19] and UCSD's WISARD [17]. Many technologies have reached commercial deployment, including the Nonin Bluetooth pulse oximeter used in our implementation and, with wireless network quality getting better and battery life getting longer, we expect to see significant adoptions of wireless medical devices in the near future. However, current systems solve only a piece of the overall problem. For example, the Nonin pulse oximeter can pair with nearby computers that use Bluetooth, but has little capability itself and, in particular, it is not able to identify the subject on which it is being used. Also, this new technology only becomes practical if there exists an efficient mechanism to handle the heterogenous nature of the system, that is requiring a modest amount of effort to manage and install various devices, potentially from different vendors. The *medical mediator* is a design that effectively addresses this gap using a mobile computer that can carry out identifications and mediate communications with devices that lack identification capabilities.

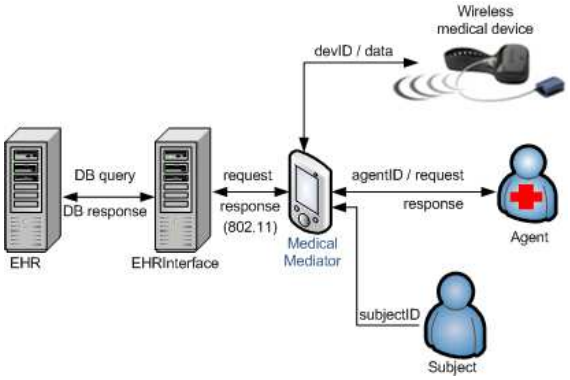

Figure 1. Communications

We have developed an overall system architecture, which has been implemented and tested with existing hardware platforms. Our tests show that such a system is feasible and could be deployed in practice. The medical mediator combines patient identification and medical data capturing functionalities into one single handheld device and provides automatic aggregation of the two. It can assist the nurse in detecting and preventing medical errors, which, if go unnoticed, might lead to harmful consequences for patients such as injuries or even death. We will discuss its architecture and intended operations in this section; we describe a prototype implemention in the next section.

Architecture. Figure 1 shows communication paths between the components and with the clinicians and subjects. There are three software components: an elec-

tronic health record database EHR, the medical mediator software, and a server which processes data and interfaces with the EHR: EHRInterface.

EHR contains information about the *agents* (clinicians like nurses), *subjects* (patients) and medical *devices*. This includes the name and role of each agent in the hospital, medical notes (such as medication allergies) of each patient, a short description of each medical device, its functionality (pulse oximeter/ scale) and its working status (working/broken), etc. This information can be queried using the ID number of the corresponding object or subject. EHR also stores information about medical measuring schedules: when it should take place, who is the patient, and what type of device should be used, etc.

The mediator, Med, runs on a handheld device that can connect to EHRInterface and to medical devices. It also has the ability to read identification tags. The mediator acts as an interface between the *participants* (agents, subjects and devices) and our backend system, which comprises EHR and EHRInterface. Each agent, subject and device has a tag containing their identification number (ID). Note that we do not limit ourselves to any particular identification mechanism. A barcode or any other type of automated ID such as RFID can be used to identify the participants.

The last component, EHRInterface, processes requests from the agent, makes queries to EHR, analyzes medical data, checks for possible errors, and commits changes to EHR based on decisions made by the agent. By comparing data given by Med and data stored in EHR, EHRInterface is capable of detecting and preventing a wide range of errors during the identification and data capture process.

System Operations. At the beginning of her shift, a nurse checks out a PDA with the mediator software installed and running. She scans her badge to associate herself with the mediator, perhaps entering a password or PIN. To take a measurement for a given patient and device the nurse scans the patient's tag and the tag of the medical device. The two IDs are sent from the mediator to EHRInterface. Then EHRInterface queries EHR to get detailed information regarding these IDs, such as the patient's name and the device's description. If EHRInterface detects any mismatch, such as a wrong combination of IDs, or no scheduled measuring for the patient at this given time, etc, an error is raised that may cancel the whole transaction and notify the nurse through the mediator; otherwise the identifications are sent back to the mediator. This check is capable of detecting many common errors, such as patients wearing wrong ID tags, or nurse taking measurement of wrong patient (perhaps having a name similar with the correct one). After this point, the nurse can verify these identifications one more time, making sure that these data reflect the actual patient and device, before advancing to the next stage.

The mediator then connects to the identified device if necessary. Since many of the new medical devices are enabled with Bluetooth, this may mean a Bluetooth link, but other communication technologies like ZigBee or infrared are also possible. In our implementation the nurse needs to select the identified Bluetooth device from a list; a more streamlined program could avoid this step. The mediator then collects the data captured by the identified medical device. The collected data will be tunneled from the mediator to EHRInterface. The mediator does not need to "understand" the semantics of the collected data since it will not analyze it; it will instead be parsed by EHRInterface. The result, in a human readable format, will be transferred back to Med. The nurse can now examine the outcome, along with meaningful identifications of the patient and the device before deciding whether the data is correct and should be recorded into the patient's EHR. This design offloads heavy computations from a PDA to a much more powerful server. Doing this not only reduces the cost of the system, but also eases management, both of which add to scalability and applicability of our system. If we were to rely on the mobile device for computational power, the mediator would probably need to be a laptop or tablet PC, which are much more expensive than a PDA. However, the more serious concern is configuration management: in order for a new medical device to be introduced into the system its software must be installed on all of the mobile devices. Even in a medium-sized hospital, this number could be hundreds. In our system only the backend server needs to be updated.

4. Med2 Implementation

We implemented a prototype of our design *Med2* and tested it in taking real medical measurements. In our implementation, we put both EHR and EHRInterface components in a single laptop equipped with Intel Core2Duo 2.00GHz processor, 2GB of RAM, running Microsoft Windows XP. The Med2 software runs on top of an HP iPaq 4150 PDA with Microsoft Windows Mobile 2003 platform. The Pocket PC has 64MB of RAM, 400MHz Intel XScale processor, and an SD slot armed with a SDIO RFID reader from SDiD. The PDA connects to the back end server via WiFi, and communicates with a Nonin 4100 wireless pulse oximeter via Bluetooth. We use ISO 15693 compliant RFID tags to identify the participants. The RFID tags in the prototype are shaped like

Figure 2. Med2 Hardware

credit cards, but these tags also can be had in wrist bands or as tags that can be stuck on devices. The mediator identifies a tag by being held within a few centimeters of it. This proximity reduces the risk that tags from other nearby devices or patients will be collected accidentally. Figure 2 shows the Med2 mediator in action.

Med2 has about 3000 lines of C++ code and a small mySQL database. This includes code for our framework, a piece of code to control the SDiD reader and read data from RFID tags, and a piece of code to interpret data from the pulse oximeter. EHRInterface was programmed using MFC while the mediator software on the PDA was developed with Microsoft .Net Compact Framework.

Figure 3 shows detailed modules of Med2. The functions of the modules are intuitive and can typically be deduced from their names. *Identification Logic* and *Data Collector* help the nurse to identify the participants, and collect readings from medical devices. *Transaction Control* at EHRInterface creates and terminates transactions. To enhance reliability of the system, each transaction is realized using a separated instance of *Transaction Logic*. This is to make sure that if anything wrong happens in a transaction, it will be contained and cannot affect other transactions taking place at the same time. Readings collected from medical devices will be interpreted by the *Device Logic* module. This is a layer of abstraction that addresses heterogeneity by providing a common interface when handling multiple types of devices from the same or different vendors. Each type of device will have its corresponding device logic implemented in a dynamic library. This logic may include wrappers for vendor-specific code to assure compatibility in the EHR or the mediator interface. This approach allows new devices to be installed with no changes to the code on the mediator and controlled changes in the EHR schemas.

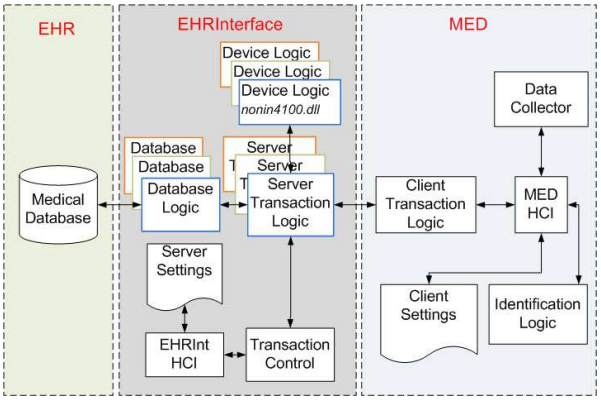

Figure 3. Med2 Software Components

We tested our implementation by taking blood oxygen-percentage and heart rate readings on people in our lab. We set up a few scenarios with errors within the specified protection envelope and showed that our implementation was able to detect and prevent these errors. Several people were given RFID cards, each with a programmed ID that identifies its owner. One of them was given role as a nurse and the others were patients. The Nonin Bluetooth Oximeter was also given an ID. These roles and IDs were reflected in our EHR database, which was implemented using mySQL. In a normal transaction, the nurse began by scanning a patient ID and the pulse oximeter ID. She then waited to receive the patient's name and picture and the device's description, and verified them. She then put the hand of the patient in the oximeter, selected it in the list of devices discovered by the PDA and waited for the data collecting process to complete. The raw data was sent to our backend system and the patient's blood oxygen percentage was sent back to the nurse. This result is displayed in HTML format, along with other identification information. The nurse approved the result and the device moved into a state ready for the next patient.

To prove the error detection capability of our system we set up a few abnormal scenarios as well. These fell into the protection envelope and all of them were detected. There were trivial errors such as scanning the wrong type of ID or using the wrong type of device. Some type of errors were even prevented before they could happen. We tried changing the status of the oximeter from working to broken, or changing the schedule to another date in our EHR database. These actions triggered our backend system to stop the trans-

action and informed the nurse of the according error.

There is work on improving patient identification using automated identification techniques [28, 15] but these works do not address aggregating automated patient identification with automated medical data capture. Due to time and resource constraints, we did not fully implement some aspects of Med2, such as security and optimized power management. But our prototype and testing results show that these issues can be addressed and the approach is feasible to deploy in practice using inexpensive stock platforms, communication cards, and medical devices.

5. Formalism

Our approach to modeling AIDC is to represent each of the environment (typically entities supplying identity and information), the human workflow, the identification system and the identification platform (typically a database such as an EHR on an inventory system) as CSP processes to be composed with synchronization. The environment and the identification platform should be thought of as fixed and the purpose of their specification is to capture existing functionality. The specifications for each of the human workflow and the identification system should be thought of as stating the required behavior to be implemented. The composition of the human workflow and the identification system is the total "system" to be supplied. However, since the human behavior can not be guaranteed, it is critical that its specification be broken out separately to clarify the precise limitations of the system deployed.

For each type of pairing of entities in the environment, human agents, identification system components and platform components, we associate a parameterized family of channels, where the parameters give the specific implicit and explicit participating entities. In our examples, we have given the parameters as indexing subscripts and superscripts. To the extent one can say one party "owns" a channel, we have used the subscript to indicate the owner. The superscripts indicate information about the other participant(s) in the communication, and the source(s) of the data being communicated. These are mathematical labeling of the channels by "real world" entities and are not visible to the system being implemented; only the data transmitted is actually visible. However, these indices play a critical role in enabling us to state the needed correspondence properties guaranteeing that the recorded data had originated from the expected source, *i.e.*, they have the expected identity. To clarify this general methodology, we will demonstrate it through the example of the medical mediator.

We explore a scenario at a hospital where nurses take patient readings from devices capable of reporting results wirelessly. The challenge here is to ensure that the correct reading gets associated with the correct patient in the hospital database. To aid in ensuring this, all elements of the hospital environment are equipped with identification tags, and the nurse is required to identify with those tags the patient and the device being used, and then the system should ensure that the data collected are associated with the identified patient and device. The whole process can be seen as an interaction among the physical actions taking place in the environment, like the nurse taking the reading using a device and the actions in the electronic system, which receives, interprets and stores the readings collected from patients.

In accordance with the general methodology outlined above, the environment comprises the patients, the medical devices, and all tags associated with these entities. The nurse must adhere to the human workflow. The medical mediator (the PDA plus readers and software) is the identification system, and the combination of the EHR and its interface make up the identification platform. To model these elements we will use the CSP processes Tag, Pat, Nurse, Med, Device, EHRInterface and EHR. Some of these processes will be further described in terms of subprocesses, such as TakeCkReading for Nurse. The channels are as follows: HCI_m^n is used for interactions between the nurse n and the medical mediator m; $\mathsf{RFIDChan}_o^{n,m}$ for communicating RFID tag numbers for a tagged object o to a medical mediator m operated by nurse n; $\mathsf{BTAddr}_d^{n,m}$ is the Blue-tooth connection between m and device d selected by n, while BTScan_m represents scanning the radio frequencies for blue-tooth devices; EHRCh^m is the channel for m to communicate with the EHR interface to get names associated with id numbers, and $\mathsf{EHRBECh}^m$ is the channel for m to use to store information in the EHR.

Human Workflow. In Section 2, we introduced the nurse's workflow, including its CSP specification. That specification relied upon the specification TakeCkReading of the workflow to be followed when taking and verifying a reading for a patient. For this phase, when the nurse is ready to take a reading he pushes a GetReading button on the medical mediator, then selects a blue-tooth device from the list displayed assuming it is present, examines the results of the reading and decides whether to store the results if they are

meaningful. In CSP, we can encode this as:

$$
\begin{array}{l}
\mathsf{TakeCkReading}(n,m,p,d) = \\
\quad \mathsf{HCI}^n_m!\mathsf{GetReading} \rightarrow \mathsf{HCI}^n_m?X \rightarrow \\
\quad \text{if } \mathsf{BTAddr}^{n,m}_d \notin X \\
\quad \text{then } \mathsf{HCI}^n_m!\mathsf{No} \rightarrow \mathsf{Skip} \\
\quad \text{else } \mathsf{HCI}^n_m!\mathsf{BTAddr}^{n,m}_d \rightarrow \mathsf{HCI}^n_m?data \rightarrow \\
\quad \text{if } data = \mathsf{Error} \\
\quad \text{then } \mathsf{HCI}^n_m!\mathsf{OK} \rightarrow \mathsf{TakeCkReading}(n,m,p,d) \\
\quad \text{else } ((\mathsf{HCI}^n_m!\mathsf{Yes} \rightarrow \mathsf{Skip}) \sqcap \\
\qquad (\mathsf{HCI}^n_m!\mathsf{No} \rightarrow \mathsf{Skip}))
\end{array}
$$

Environment. The tags in the environment should be modeled as processes that repeatedly announce their identification number to any party listening. All we directly know about a patient is her tag. A blue-tooth device is similar to a tag, except that it chooses a fresh set of data for each communication, instead of a single id number. A device is viewed as a combination of a tag and a blue-tooth device. Separate from the individual blue-tooth devices, we will have a channel that is used for detecting all available blue-tooth devices. These are specified by:

$$
\begin{array}{l}
\mathsf{Tag}(o) = \underset{n,m}{\Box}\, \mathsf{RFIDChan}^{n,m}_o!(\mathsf{IDnum}(o)) \rightarrow \mathsf{Tag}(o) \\
\mathsf{Pat}(p) = \mathsf{Tag}(p) \\
\mathsf{BTDev}(d) = \underset{n,m}{\Box}\, (\underset{data}{\sqcap}\, (\mathsf{BTAddr}^{n,m}_d!data \rightarrow \mathsf{BTDev}(d))) \\
\mathsf{Device}(d) = \mathsf{Tag}(d) \,|\!|\!|\, \mathsf{BTDev}(d) \\
\mathsf{BTDevs} = \underset{m}{\Box}(\underset{X \subseteq \mathsf{BTDevices}}{\sqcap} \mathsf{BTScan}_m!X \rightarrow \mathsf{BTDevs})
\end{array}
$$

Identification Platform. The two database components that are relied upon by the medical mediator are the EHR and the EHR interface. For the interface, we assume that it can somehow determine if the tag ids make sense, sort them and accompany them with the corresponding names. The EHR is just required to accept whatever it is sent.

$$
\begin{array}{l}
\mathsf{EHRInterface}(m) = \mathsf{EHRCh}^m?x \rightarrow \\
\quad ((\mathsf{EHRCh}^m!\mathsf{Error} \rightarrow \mathsf{Skip}) \sqcap \\
\quad (\sqcap \left\{ \begin{array}{l} (n_1,n_2) | \exists y_1\, y_2 . (y_1,y_2) = x \vee (y_2,y_1) = x \\ \quad \wedge (n_1 = \mathsf{Name}(y_1) \wedge n_2 \mathsf{Name}(y_2)) \end{array} \right\} \\
\qquad \mathsf{EHRCh}^m!n_1 \rightarrow \mathsf{EHRCh}^m!n_2 \rightarrow \\
\qquad \mathsf{EHRCh}^m?r \rightarrow \mathsf{EHRCh}^m!FormatRaw(r) \rightarrow \\
\qquad \mathsf{EHRCh}^m?z \rightarrow \text{ if } z = \mathsf{Auth}(n,m) \\
\qquad \text{then } (\mathsf{EHRBECh}^m!(n,m,y_1,y_2,EHR(r)) \rightarrow \mathsf{Skip}) \\
\qquad \text{else } \mathsf{Skip}))
\end{array}
$$

$$
\mathsf{EHR} = \underset{m}{\Box}(\mathsf{EHRBECh}^m?x \rightarrow \mathsf{EHR})
$$

Identification System. The medical mediator was described in considerable detail in Sections 3 and 4. We give the high-level CSP specification here.

$$
\begin{array}{l}
\mathsf{Med}(n,m) = \mathsf{HCI}^n_m?\mathsf{GetID} \rightarrow \mathsf{HCI}^n_m?x_1 \rightarrow x_1?y_1 \rightarrow \\
\quad \mathsf{HCI}^n_m?x_2 \rightarrow x_2?y_2 \rightarrow \mathsf{EHRCh}^m!(y_1,y_2) \rightarrow \mathsf{EHRCh}^m?n_1 \\
\quad \rightarrow \text{ if } n_1 = \mathsf{Error} \text{ then } \mathsf{HCI}^n_m!n_1 \rightarrow \mathsf{HCI}^n_m?\mathsf{OK} \rightarrow \mathsf{Skip} \\
\quad \text{else } \mathsf{EHRCh}^m?n_2 \rightarrow \mathsf{HCI}^n_m!(n_1,n_2) \rightarrow \mathsf{HCI}^n_m?z \rightarrow \\
\quad \text{if } z = \mathsf{Yes} \text{ then } \mathsf{MedRead}(n,m) \text{ else } \mathsf{Skip} \\
\mathsf{MedRead}(n,m) = \mathsf{HCI}^n_m?\mathsf{GetReading} \rightarrow \\
\quad \mathsf{BTScan}_m?X \rightarrow \mathsf{HCI}^n_m!X \rightarrow \mathsf{HCI}^n_m?y \rightarrow \\
\quad \text{if } y = \mathsf{No} \text{ then } \mathsf{Skip} \\
\quad \text{else } y?rdata \rightarrow \mathsf{EHRCh}^m!rdata \rightarrow \mathsf{EHRCh}^m?fdata \rightarrow \\
\quad \mathsf{HCI}^n_m!fdata \rightarrow \text{ if } fdata = \mathsf{Error} \\
\quad \text{then } \mathsf{HCI}^n_m?\mathsf{OK} \rightarrow \mathsf{MedRead}(n,m) \\
\quad \text{else } \mathsf{HCI}^n_m?z \rightarrow \text{ if } z = \mathsf{Yes} \\
\quad \text{then } \mathsf{EHRCh}^m!\mathsf{Auth}(n,m) \rightarrow \mathsf{Skip} \\
\quad \text{else } \mathsf{Skip}
\end{array}
$$

Protection Envelope. A nurse may unwittingly make mistakes while following the suggested workflow of hospital AIDC. A simple example can be the case when the nurse is unsure as to whether he has scanned the patient's ID successfully and scans it again. This should not cause any serious compromise of the workflow and life will be easier for the nurse if he is not required to restart the whole process if he scans an identity multiple times. Hence, it is felt that there should be a more flexible workflow accepted by the hospital AIDC system than the workflow suggested to the nurses. This more flexible workflow will form a protection envelop around the suggested workflow and help the nurses to be on track in spite of mistakes unwittingly made.

In a protected workflow for the nurse, multiple scans of the identity tag of a patient or a device has the same effect as scanning it once. After a nurse has scanned a patient, a nondeterministic internal choice is made as to whether the patient will be scanned again or whether the device will be scanned as the next action. Another possible situation arises when the nurse scans a medical device's tag first and then scans a patient's tag instead of scanning the tag of the patient first and then the device's tag. The more flexible workflow can look like this:

$$
\begin{array}{l}
\mathsf{NursePE}(n,m) = \mathsf{HCI}^n_m!\mathsf{GetID} \rightarrow \\
\quad ((\underset{p}{\sqcap}\mathsf{ScanPat}(n,m,p)) \sqcap (\underset{d}{\sqcap}\mathsf{ScanDev}(n,m,d))) \\
\mathsf{ScanPat}(n,m,p) = \mathsf{HCI}^n_m!(\mathsf{RFIDChan}^{n,m}_p) \rightarrow \\
\quad (\mathsf{ScanPat}(n,m,p) \sqcap (\underset{d}{\sqcap}(\mathsf{HCI}^n_m!(\mathsf{RFIDChan}^{n,m}_d) \rightarrow \\
\quad \mathsf{SeeID}(n,m,p,d)))) \\
\mathsf{ScanDev}(n,m,d) = \mathsf{HCI}^n_m!(\mathsf{RFIDChan}^{n,m}_d) \rightarrow \\
\quad (\mathsf{ScanDev}(n,m,d) \sqcap (\underset{p}{\sqcap}(\mathsf{HCI}^n_m!(\mathsf{RFIDChan}^{n,m}_p) \rightarrow \\
\quad \mathsf{SeeID}(n,m,p,d))))
\end{array}
$$

$$
\begin{array}{l}
\mathsf{SeeID}(n,m,p,d) = \mathsf{HCI}^n_m?x \rightarrow \\
\quad \text{if } x = (\mathsf{Name}(p), \mathsf{Name}(d)) \\
\quad \text{then } ((\mathsf{HCI}^n_m!\mathsf{Yes} \rightarrow \mathsf{TakeCkReading}(n,m,p,d)) \\
\quad \text{else if } x = \mathsf{Error} \text{ then } (\mathsf{HCI}^n_m!\mathsf{OK} \rightarrow \mathsf{Skip}) \\
\quad \text{else } (\mathsf{HCI}^n_m!\mathsf{No} \rightarrow \mathsf{Skip}))
\end{array}
$$

6. Safety Properties

There are a variety of different semantics and models for CSP processes. We concentrate on the trace model and the failure model in this work. In a trace model a process is represented by the sequences of communications or events it can perform. In the stable failures model, a process is represented by its failures. A failure for a process, P, is given by a pair (s, X), where s is a trace of P and X is a set of things it can refuse to perform after performing s. We intend to provide some desired safety and liveness properties for the hospital AIDC system. We will provide the specifications for a CSP process by providing the most nondeterministic process satisfying the specification. The CSP process for a hospital AIDC system satisfies a specification S if the CSP process P representing the specification is refined by the AIDC system process. We will consider two different types of refinement: trace refinement for safety properties and failure refinement for liveness properties [21]. A process Q is a trace refinement of another process P, denoted by $P \sqsubseteq_T Q$ if $\mathsf{traces}(Q) \subseteq \mathsf{traces}(P)$. A process Q is a failure refinement of another process P, denoted by $P \sqsubseteq_F Q$ if $\mathsf{failures}(Q) \subseteq_F \mathsf{failures}(P)$. If we have a process P and its safety specification S and we have $S \sqsubseteq_T P$, then we can say P is a safe process, as all its traces are traces of the specification, so no bad event can be performed by P. Similarly, failure refinement can be used to express liveness properties.

We now express the environment of a hospital AIDC system as the collection of all the identification tags representing the devices and patients and the devices sending out the readings for patients, together with the electronic records system. These are the "given" components, which we cannot control.

$$\begin{aligned}\mathsf{Given} = \quad & (\underset{d \in \mathsf{Dev}}{\|} \mathsf{Dev}(d)) \,\|\, (\underset{p \in \mathsf{Pat}}{\|} \mathsf{Pat}(p)) \\ & \| \,\mathsf{BTDevs} \,\| \,\mathsf{EHR}\end{aligned}$$

We shall refer to the combination of all instances of nurses and medical mediators as the system. It is important that only one nurse have use a medical mediator at a time. This is reflected in our specification by allowing each medical mediator to "choose" a nurse.

$$\begin{aligned}\mathsf{System} = \quad & ((\underset{m}{\|} (\underset{n}{\sqcap} (\mathsf{Nurse}(n,m) \,[\![\{\{y | \exists x. y = \mathsf{HCI}^n_m.x\}\}]\!] \\ & \mathsf{Med}(n,m)) \,[\![\{\{y | \exists x. y = \mathsf{EHRCh}_m.x\}\}]\!] \\ & \mathsf{EHRInterface})) \setminus \{y | \forall n\ m\ p\ d\ x. \\ & y \neq \mathsf{RFIDChan}^n_p.x \wedge y \neq \mathsf{EHRBECh}^m.x \wedge \\ & y \neq \mathsf{BTAddr}^{n,m}_d.x \ \wedge y \neq \mathsf{BTScan}_m.x\}); \\ & \mathsf{System}\end{aligned}$$

In the system, we hide away almost all internal and inter-element communication. The only events that we are interested in are the actual scanning of patient and device identifier tags and the ultimate sending of the medical measurement to the EHR for storage after it has been verified by the nurse.

We now provide the safety specification for the hospital AIDC system, where the specification ensures that if a tuple of data is entered into a database, stating that a particular device, d was used to take a reading x from a patient p by a nurse n, then it had been the case that the nurse n *did* identify p and d and *had* taken a medical measurement x from the patient p using d after identifying them.

$$\begin{aligned}\mathsf{Safety} = \quad & \underset{m}{\|} (\underset{n\ p\ d}{\sqcap\sqcap\sqcap}(((\mathsf{RFIDChan}^{n,m}_p.\mathsf{IDnum}(p))^+ \rightarrow \\ & (\mathsf{RFIDChan}^{n,m}_d.\mathsf{IDnum}(d))^+) \\ & \sqcap((\mathsf{RFIDChan}^{n,m}_d.\mathsf{IDnum}(d))^+ \rightarrow \\ & (\mathsf{RFIDChan}^{n,m}_p.\mathsf{IDnum}(p))^+) \\ & \rightarrow (\mathsf{Skip} \sqcap (\mathsf{BTScan}_m?X \rightarrow (\mathsf{Skip} \sqcap \\ & \quad (\mathsf{if}\ \mathsf{BTAddr}^{n,m}_d \in X\ \mathsf{then}\ (\mathsf{BTAddr}^{n,m}_d?r \rightarrow (\mathsf{Skip} \sqcap \\ & \quad \mathsf{EHRBECh}^m.(n, m, \mathsf{IDnum}(p), \mathsf{IDnum}(d), EHR(r)) \\ & \quad \rightarrow \mathsf{Skip}))\,\mathsf{else}\,\mathsf{Skip}))))));\mathsf{Safety}\end{aligned}$$

Now, we are ready to specify the desired safety and liveness properties. Let $\mathsf{Vis} = \{y.(\exists n\ d\ x.\ y = \mathsf{RFIDChan}^{n,m}_d.x) \vee (\exists m\ z.\ y = \mathsf{EHRBECh}^m.z)\}$. The properties then are as follows:

$$\begin{aligned}& \mathsf{Safety} [\![\{\mathsf{Vis}\}]\!] \mathsf{Given} \sqsubseteq_T \mathsf{System} [\![\{\mathsf{Vis}\}]\!] \mathsf{Given} \\ & \mathsf{LIVE} \sqsubseteq_F \mathsf{System} [\![\{\mathsf{Vis}\}]\!] \mathsf{Given}\end{aligned}$$

where $\mathsf{LIVE} = \underset{x}{\sqcap} x \rightarrow \mathsf{LIVE}$. LIVE is the most nondeterministic process that is deadlock free. The first property states that no wrong, random, non-associated tuple gets stored in EHR and the second property states that the nurse can eventually take steps and will not get deadlocked.

We have sketched proofs for the above results based on their semantics using traces of events. Due to the large number of cases to be considered, we feel that such a proof is easily subject to accidental omissions and hence is best checked by automated assistant. To this end we have encoded the CSP processes described in this paper two such tools. The first system we used was the fully automated CSP failures-refined prover FDR2 (Failures-Divergence Refinement) [1]. Our experience with this system is limited, but a straightforward encoding led the system to run for hours, probably indicating divergence. The second system we used was the extension CSP-prover [13] to the interactive theorem prover Isabelle [18]. In this system, there is a restricted type system for communications and the encoding entailed creating a number of functions for coercing data into this type system. Also, certain operators were lacking

(index interleaving, in particular), and so we had to expand the code to accommodate this lack. As of this writing the proofs in CSP-prover of the above two results are incomplete. We conclude from this effort that there are interesting challenges to automating proofs of AIDC workflows such as the ones for the mediator.

To summarize these results in less technical language, there are three possibilities for the outcome of operator actions. First there are actions that make progress by providing proper updates of the EHR. This will occur when the nurse adheres to the specified workflow. Second there are actions that lie within the protection envelope but do not make progress. For example, repeated reading of a patient's tag is tolerated by the system although it is not the specified workflow. A detected error occurs when reading tags on two devices (rather than on a patient and a device). Third there are actions that corrupt the EHR database. These require significant deviation from specified workflow such as reading the tag on one patient and then collecting a reading from a different patient.

7. Related Work

We are not aware of any prior work on formal modeling and analysis of ADIC workflows in health care, but formal techniques have been applied to workflows in general, there have been case studies in heath care applications, and there is a significant body of non-formal work on AIDC in health care.

Yong *et. al.* in [27] use CSP to model and refine workflows to prove correctness properties. Puhlmann *et. al.* [20] use the π-calculus to model workflow patterns. Stafansen [23] shows how the Calculus of Communicating Systems (CCS) can be used to model workflow. Davulcu *et. al.* [8] use Concurrent Transaction Logic (CTR) to specify, analyze, and schedule workflows. Van der Alast *et. al.* [24] use Petri nets to model workflow. Petri nets are able to model Yet Another Workflow Language(YAWL), which in turn, can model a wide range of workflow patterns [25].

Barkaoui *et al* [5] use Petri nets to model, verify and improve an operating room workflow. Song *et al* [22] provide a classification of computer-aided health care workflows and workflow application issues. Van der Alast *et al* [26] describes a method for finding workflows as Petri nets automatically from hospital process logs. Ardissono *et al* [3] present a framework for the management of context-aware workflow systems. They use the framework to develop a prototype application handling medical guidelines specifying workflow for monitoring patients treated with blood thinners. In our research on AIDC we attempted to use π-calculus and Finite State Machines (FSMs) before concluding that CSP was better-suited to our needs because of its algebraic notation, simple treatment of communication, and its representation for internal and external choice. However, it is likely other models discussed above could provide good formalisms for reasoning about AIDC.

AIDC techniques are now being used in many different areas in the health care sector. One notable area is the use of AIDC techniques for the blood management to reduce blood handling errors so that the chances of bad blood transfusions is reduced. Clarke *et al* [7] model and verify an in-patient blood transfusion process. Al Nahas *et al* [16] overview research in various applications of RFID-based AIDC systems in hospitals. Automated identification of patients [4] provides more accurate and efficient association of patients with their medical readings and the medical procedures they should undergo. AIDC techniques are being deployed to provide error-free and fully-audited drug management in hospitals [4, 12]. Hospital-wide AIDC [9, 10] is being investigated for tracking patients, assets, and staff.

8. Conclusion

We have introduced a methodology for modeling and analyzing AIDC workflows to provide high assurance protection envelopes. Such techniques will aid the development of computer and networking systems that support AIDC in safety-critical workflows like those used in hospitals. To illustrate and demonstrate the value of our approach we have designed and implemented a medical mediator system which provides RFID identification using a mobile device that links to medical devices to collect patient readings and enters them automatically in the EHR of the (correct) patient within well-specified and verified conditions. This is the first work on formal analysis of AIDC workflows; it contributes insight into good primitives for clear specifications in this area and promises wide applicability.

Acknowledgements. This work was supported in part by NSF CNS 07-16626, NSF CNS 07-16421, NSF CNS 05-24695, ONR N00014-08-1-0248, NSF CNS 05-24516, DHS 2006-CS-001-000001, and grants from the MacAruthur Foundation and Boeing Corporation. Anh Nguyen was supported by a fellowship from the Vietnam Education Foundation. The views expressed are those of the authors only.

References

[1] FDR2(Failures-Divergence Refinement). internet website: `http://www.fsel.com/`.

[2] Transportstion Security Administration, US Department of Homeland Security. internet website, www.tsa.gov.

[3] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. Context-aware workflow management. In L. Baresi, P. Fraternali, and G.-J. Houben, editors, *ICWE*, volume 4607 of *Lecture Notes in Computer Science*, pages 47–52. Springer, 2007.

[4] J. Bardram. Applications of context-aware computing in hospital work: examples and design principles. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, *SAC*, pages 1574–1579. ACM, 2004.

[5] K. Barkaoui, P. Dechambre, and R. Hachicha. Verification and optimisation of an operating room workflow. In *HICSS*, page 210, 2002.

[6] M. R. Chassin and E. C. Becher. The Wrong Patient. *Ann Intern Med*, 136(11):826–833, 2002.

[7] L. A. Clarke, Y. Chen, G. S. Avrunin, B. Chen, R. L. Cobleigh, K. Frederick, E. A. Henneman, and L. J. Osterweil. Process programming to support medical safety: A case study on blood transfusion. In M. Li, B. W. Boehm, and L. J. Osterweil, editors, *ISPW*, volume 3840 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 2005.

[8] H. Davulcu, M. Kifer, C. R. Ramakrishnan, and I. V. Ramakrishnan. Logic based modeling and analysis of workflows. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington*, pages 25–33. ACM Press, 1998.

[9] P. Fuhrer and D. Guinard. Building a smart hospital using rfid technologies. In H. Stormer, A. Meier, and M. Schumacher, editors, *ECEH*, volume 91 of *LNI*, pages 131–142. GI, 2006.

[10] J. Halamka. Early experiences with positive patient identification. journal of healthcare information management. *Inf. Syst.*, 20(1):25–27, 2006.

[11] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[12] B. Houliston. Integrating RFID technology into a drug administration system. In *Bulletin of Applied Computing and Information Technology*, 2005.

[13] Y. Isobe1 and M. Roggenbach. CSP-Prover. http://staff.aist.go.jp/y-isobe/CSP-Prover/CSP-Prover.html.

[14] J. M. C. Kohn, L. T. and M. S. Donaldson. To err is human: Building a safer health system. http://www.nap.edu/catalog/9728.html.

[15] C.-L. Lai, S.-W. Chien, L.-H. Chang, S.-C. Chen, and K. Fang. Enhancing medication safety and healthcare for inpatients using rfid. *Management of Engineering and Technology, Portland International Center for*, pages 2783–2790, 5-9 Aug. 2007.

[16] H. A. Nahas and J. S. Deogun. Radio frequency identification applications in smart hospitals. In *CBMS*, pages 337–342. IEEE Computer Society, 2007.

[17] U. of California at San Diego. Wireless internet information system for medical response in disasters. https://www.wiisard.org/.

[18] U. of Cambridge and T. U. Mnchen. Isabelle . http://isabelle.in.tum.de/index.html.

[19] S. of Engineering and A. S. H. University. Codeblue: Wireless sensor networks for medical care. http://www.eecs.harvard.edu/mdw/proj/codeblue/.

[20] F. Puhlmann and M. Weske. Using the π-calculus for formalizing workflow patterns. In W. M. P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management*, volume 3649, pages 153–168, 2005.

[21] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.

[22] X. Song, B. Hwong, G. Matos, A. Rudorfer, C. Nelson, M. Han, and A. Girenkov. Understanding requirements for computer-aided healthcare workflows: experiences and challenges. In L. J. Osterweil, H. D. Rombach, and M. L. Soffa, editors, *ICSE*, pages 930–934. ACM, 2006.

[23] C. Stefansen. Smawl: A small workflow language based on ccs. In O. Belo, J. Eder, J. F. e Cunha, and O. Pastor, editors, *CAiSE Short Paper Proceedings*, volume 161 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[24] W. M. P. van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *ICATPN*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 1997.

[25] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[26] W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[27] P. Y. H. Wong and J. Gibbons. A process-algebraic approach to workflow specification and refinement. In M. Lumpe and W. Vanderperren, editors, *Software Composition*, volume 4829 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 2007.

[28] W. Yu and A. Ramani. Design and implementation of a personal mobile medical assistant. *Enterprise networking and Computing in Healthcare Industry, 2005. HEALTHCOM 2005. Proceedings of 7th International Workshop on*, pages 172–178, 23-25 June 2005.