Model Checking for Verification of Interactive Health IT Systems

Keith A. Butler, PhD1, Eric Mercer, PhD2, Ali Bahrami, PhD3, Cui Tao, PhD4
1University of Washington, Seattle, WA; 2Brigham Young University, Provo, UT, 3Medico Systems, Bellevue, WA, 4University of Texas, Houston, TX

Abstract

Rigorous methods for design and verification of health IT systems have lagged far behind their proliferation. The inherent technical complexity of healthcare, combined with the added complexity of health information technology makes their resulting behavior unpredictable and introduces serious risk. We propose to mitigate this risk by formalizing the relationship between HIT and the conceptual work that increasingly typifies modern care. We introduce new techniques for modeling clinical workflows and the conceptual products within them that allow established, powerful modeling checking technology to be applied to interactive health IT systems. The new capability can evaluate the workflows of a new HIT system performed by clinicians and computers to improve safety and reliability. We demonstrate the method on a patient contact system to demonstrate model checking is effective for interactive systems and that much of it can be automated.

Introduction

The importance of interactive computer systems in healthcare has grown rapidly with the proliferation of networks of personal devices, laptops, and desktops. Rigorous methods for design and verification, however, have lagged far behind. Standard methods, such as usability testing, that were adequate for interactive systems used in business offices and homes, were never intended for safety-critical domains. The inherent technical complexity of healthcare, combined with the added complexity of health information technology (HIT) makes their resulting behavior unpredictable and introduces serious risks. These risks threaten patient safety, disrupt care, undermine the added value of HIT, and impede its adoption.

Model checking is one technology that has great potential to help mitigate the risks of unpredictability in complex safety-critical systems. A model checker proves whether a subject system satisfies the properties that it needs to meet in every possible behavior of the system. Applying model checking directly to interactive systems that are employed by clinicians, however, presents distinct challenges. One of the most severe is the difficulty to define clearly the conceptual work, such as disease diagnoses, treatment plans, surveillance, case-management, hospital admissions, and schedules for scarce equipment that is common in modern healthcare. Clinical workflow studies have shown these types of conceptual work products to be fundamental in HIT and care.

Until recently the problem of how to specify conceptual work products in a sufficiently rigorous manner prevented model checking of interactive systems for this aspect of care. Theoretical advances in cognitive science combined with standards for declarative knowledge modeling now offer a technique to clearly specify conceptual work products (CWP), which in turn enables model checking for interactive HIT systems. The technique, presented in this paper, builds on earlier work that innovated CWP specifications as reference models to guide the design of interactive systems with workflows performed by human and machine agents. Early successes solved difficult design problems in aircraft schedules that integrate flying missions with aircraft maintenance, online self-support for users of personal computers with software problems, case management for MS outpatients, and NASA’s planning system to coordinate maneuvers of the International Space Station.

We introduce a new evaluation criterion for model checking interactive HIT systems using the CWP it is supposed to produce. A CWP can be specified using standard domain modeling techniques for a clear description of a valid starting state, a valid ending state, and intermediate states that describe its acceptable evolutions. Importantly, the CWP abstraction tells us what must be accomplished by an HIT system independently of how it is accomplished. Rather than checking the state of a machine, the focus is on the state of the CWP as clinicians and their computing devices transform it to its goal state. Intuitively, this new approach shifts verification focus to the work intended to be accomplished by an interactive HIT system by proving that it accomplishes the work declared in the CWP.

To date, much of the research with model checking in health care has focused on formalizing workflows and proving temporal properties on workflows or on developing new modeling languages to describe workflows and human machine teaming in those workflows. A key principle of the new approach is that the CWP has states that are defined by combinations of attribute values that are independent of the means to attain them in the
workflow. This allows established model checking to be extended to verify workflows that have both human-performed tasks and machine-performed functions with vastly different capabilities in the same model. This new capability has potential to bring new levels of safety and reliability to HIT systems.

In the remainder of this paper we give an overview of the new method, illustrate the new techniques for modeling workflow and CWP with a small, interactive system, and describe a proof-of-technology study of the new model checking approach. We conclude by discussing the benefits to safety of HIT and to software engineering more generally, and outlining the next steps for research.

Overview of the New Method
We conducted a study of how to extend established model checking technology to exploit CWP as the evaluation criterion. Our objective was to determine whether CWP could serve as the evaluation criterion without making major changes to model checking technology. Our initial study assumed normative performance on all tasks by both human and machine performers. Our rationale was if a system could not check out under ideal conditions it did not merit further checking until it was improved. The functional components of the new approach in the study are shown in figure 1.

As shown in figure 1 the components have two main groups: Product & Process Models of the interactive system on the left, and the Verification Components on the right. The components for verification are widely recognized as well-established in effective practice for electronic and for software such as distributed systems. These verification components prove whether a transition system implements a specification.

The Process & Product Models on the left are two complementary innovations: for representing the products of the conceptual work of clinical care; and the workflow processes performed by clinicians and computers that create those products.

1. MATHflow is a graphical process diagramming tool that integrates models of clinical workflow with the use and change of information resources. There is growing interest in process models of workflow to analyze and design HIT. MATHflow is a partial implementation and an extension of the OMG’s recent standard for Business Process Modeling Notation (BPMN), which was created to support the development of software requirements for people working in groups that are supported by computing. As shown in figure 4 the MATHflow extension adds a properties editor for each task in a workflow to enter attributes to represent the information the task uses, the containing resource where each was accessed, and the destination information resource if the task changes any attribute values. The associations among tasks, attributes, and information resources are automatically recorded in MATHflow’s information dictionary.

2. In the terminology of knowledge representation a CWP is a declarative specification, as opposed to procedural specification of how-to do something. Rummelhart and Norman were among the first to report scientific
evidence that knowledge of what is learned and stored differently in human memory than the knowledge of how-to do things. CWP complement and complete procedural workflow models of care by making their purpose clearer and more meaningful. Generally, a workflow is constrained by the product it is supposed to produce because it implies so much about the activities and resources that are needed in the workflow. In the terminology of BPMN a CWP is a specification of a token, i.e., the entity that flows through a workflow process of task activities that operate on it to change its state. We use the term “conceptual work product” to mean the entity as it evolves through all its intermediate states to reach its goal state. In task-analytic methods such as TURF, a CWP corresponds to the work domain ontology. Complex CWP may require formal ontology modeling languages, such as OWL, while simpler CWP can be specified with standard declarative modeling languages such as a UML class diagram. In methods that use a domain model a CWP would be a key part of it. Figure 5 shows the class and state diagrams of the CWP for the proof-of-technology study.

The Subject System
The interactive system selected for model checking was Priority Contact for doctor-patient communication on test results and treatment plans for chronic disease. When patients have complex conditions doctors need to have real-time discussions with them about test results, diagnoses, and treatment options. Only about 10% of patients in this health system use internet, so phone is the modality of choice. Doctors also use the discussion to determine whether patients who are already dealing with complex treatments and medications will be able to follow the plan. An earlier version of Priority Contact is described in detail in Butler, et al.

Priority Contact illustrates the benefits of information technology for telemedicine that integrates desktop with phone technology, and the participation of clinician and patient. It also illustrates the liabilities of systems that involve remote, asynchronous participation that is supported by computing. The Priority Contact system depends on people and computing agents performing their tasks without the visibility that is more available when they are done manually. If any should fail there could be risk to patient health and safety that might go undetected.

The Workflows of Priority Contact
Figure 2 shows a screen shot of the MATHflow model of Priority Contact workflows that correspond to four levels of urgency, which were selected by the doctor in a previous task to initiate a contact plan. The workflows were designed to carry out their respective contact plans for:
1. The patient has a life-threatening condition - immediate contact must be made with the patient,
2. A life-changing diagnosis - a real time discussion should take place within a few days to begin major treatment,
3. A routine change to treatment - a telephone appointment should be offered to the patient to take place within fourteen days,
4. No change in treatment is required- the current plan should continue so a discussion appointment is optional to informing the patient of the test results.

MATHflow uses standard BPMN shapes to model the workflows of the four level of urgency with flows (arrows), decision gates (diamonds), tasks (plain rectangles) and sub-processes (rectangles with arrow head and cross at the bottom.
As shown in figure 3, sub-processes can, in turn, be opened to contain workflows of the same shapes. Figure 3 shows the contents of the sub-process Make Phone Clinic Appointment P3 that are displayed when it is opened. In addition, the workflow has swim lanes for graphically representing the task responsibilities of Patient and of Priority Contact as they interact to schedule a date and time for a telephone appointment for the patient and the doctor. The performer of a task is also recorded as one of its properties.
Figure 3: Modeling Human and Computer Interaction

Figure 4: Modeling the Use and Change of Information Resources
Figure 4 shows how the use and change of information is modeled as a property of the task PC writes timeDate to doctor’s schedule P3. The editor on the left is to record the information attributes that are used by the task, and the information object where it was accessed, e.g., patientName was accessed in the object Contact Plan. On the right a similar editor records any attribute whose value was changed and the destination object. In this example the task resulted in a value for confirmedAppointmentDateTime being entered to the Physician schedule object.

The MATHflow language treats information as a resource that is used or changed in tasks. In this manner any type of agent that performs a task, whether human or machine, can use information or change its value. This feature of MATHflow allows tasks to change the values of attributes that define the states of a CWP, regardless of whether they are performed by human or machine. This is a key principle for model checking on interactive systems whose workflows have both human and machine performers. The use and change of information in tasks is automatically recorded in MATHflow’s information dictionary. As a MATHflow model is incrementally developed the dictionary stores the associations among tasks, information attributes, values, and the objects where information is accessed.

The Conceptual Work Product of Priority Contact
As shown in figure 5 the CWP of Priority Contact is a Contact Plan represented with two diagrams. At the top the Contact Plan object is modeled as a UML class diagram. The bottom diagram is a UML state diagram.

![Priority Contact's Conceptual Work Product](image)

Figure 5: Priority Contact's Conceptual Work Product
Together, they are sufficient to specify the CWP and its states that the workflow must accomplish. The specification is independent of any given workflow, technology, or even cognitive strategy. Thus it can be used to verify other versions of the system that make the claim of satisfying the same requirements.

The Contact Plan is made up of several objects. At the center is a Conversation that is needed between Doctor and Patient. The Patient has Disease. The Doctor has Treatment Plan based on the Test Report. The Conversation is about the Treatment Plan. The model includes not only key attributes that define part of the actual state of the contact plan such as priority, launchTime, and resolveTime, but also doctor information, patient information, disease, treatment plan, etc. that is necessary information to accomplish the Contact Plan. All of this auxiliary information is read-only, and does not distinguish the actual state of the CWP at any given point in time and as such that information, with its various relationships to other information, can be abstracted in the verification model.

The states in the diagram at the bottom of figure 5 are defined by rules that use the key attributes. For example, the Contact Plan is in the Launched state when the attributes launchDate and launchTime have non-null values, but all other key attributes are still null. The Contact Plan can reach the Resolved state by either Conversation in progress for priorities 1-2, or Appointment scheduled for priorities 3-4. A Contact Plan may reach No longer needed if the patient somehow is no longer inMyCare of the Doctor.

**Model Checking**

The use of CWP for verifying interactive HIT systems begins by modeling the interactions of the clinician, patients, and Priority Contact as workflows in the MATHflow tool. The state diagram in the CWP is the actual specification for the interactive HIT. It defines the legal evolutions of the CWP from its initial state to its allowed final state. Any workflow must follow these legal evolutions when updating the state of the CWP. Model checking then follows every possible path in the workflow looking for bad sequences that do not reach one of the satisfactory sequences of the states that are required in the CWP specification. Sequences that violate the specification are considered unsafe because the planned progression of care does not happen, and it may go unnoticed. The rest of this section better defines model checking and describes in more detail the results of model checking the Priority Contact system. Model checking works by constructing an exhaustive proof to show that a transition system implements a specification. Modeling any system requires the definition of all events in the system; these events are sometimes referred to as atomic propositions. A model produces a set of sequences of atomic propositions that represent the behavior of the system. Typically the model is described in some high-level language, and the semantics of that language yields sequences of atomic propositions that represent the language of the model as shown in the Venn diagram in Figure 6.

![Venn diagram](image)

**Figure 6**: Venn diagram illustrating model checking where the universe is sequences of atomic propositions in the model.

The specification is also a language over atomic propositions that precisely describes sequences of interest. This language is usually presented in logic `cite{LTL, CTL}`, and that logic describes sequences of atomic propositions that represent the correct language for the system. The inversion of this language creates a language of bad sequences that are not allowed by the specification. In the Venn diagram, this is the region labeled Universe minus Specification.

Model checking is the process of proving that the intersection of the language defined by the model and the language defined by the inverse of the specification is empty: no words are common to the two languages. If such is
not the case, then model checking provides a counter-example: a sequence produced by the model but not allowed by the specification.

The use of CWP for verifying interactive HIT systems is straightforward conceptually. The interactions of clinician, patient, and Priority Contact are modeled as a workflow in the MATHflow tool. The CWP is the specification. The atomic propositions for the language of the model and specification are the state labels in the CWP state diagram. For the Priority Contact model, these are: Launched, Conversation in Progress, Appointment Scheduled, Resolved, and No longer needed. The workflow model evolves the CWP from the initial Launched state and thus produces sequences of atomic propositions. These sequences are the language of the model. Any sequence from the initial state produced on any path in the workflow is part of that language.

The specification contains those sequences allowed by the state diagram in the CWP that start in the Launched state and end in the No longer needed state and only follow transitions defined in the state diagram. Sequences not in the language of the specification represent bad sentences that the model must never produce. For the priority contact example, bad sequences are those that do not end in the No longer needed state or those that make illegal transitions such as moving from the Launched state directly to the Resolved state.

Model checking follows every possible path in the workflow looking for bad sequences. If such a sentence exists, then the language generated by the model is not contained within the language defined by the specification, and the discovered bad sentence is reported as a violation. In general, model checking does not enumerate all bad sequences (if any exist) as only one is required to prove the relationship between the language of the model and the language of the specification.

The SPIN model checker is used in this work. The translation of the CWP and MATHflow model to the SPIN input language is not explained here but there is a fairly direct mapping from MATHflow tasks to processes in Promela. Most of the complexity in the translation is in sequencing and synchronizing the processes. The output from SPIN when no error is found is a coverage summary of the input model that indicates parts of the model that are not activated in the model checking. This inactivation is a result of not providing sufficient input to the model. For example, in the Priority Contact if a P4 no change lab result is never generated, then that portion of the model is not explored in model checking and is reported as uncovered. The output form SPIN when an error is found is a trace through the input file and a trail file that SPIN is able to replay in simulation mode for debugging purposes. If desired, then SPIN can be configured to return all errors and not just the first discovered error.

The rest of this section is devoted to the results of model checking the Priority Contact example. The SPIN model checker discovered errors in the workflows of all of the priorities. To bring this result into context, only one of the errors was seeded purposely by the modelers. The other workflows were believed to be correct.

Figure 2 is the top-level workflow of the different levels of priority in the Priority Contact system. Using this figure as a reference, a brief description of the discovered errors follows:

- P1 Life Threatening: in the PC carry out contact plan P1 process a gateway had no outgoing edge and was essentially disconnected from the rest of the model. As a result, any path leading to that gateway left the CWP in an invalid end state. The error was corrected by connecting the gateway to the remaining flows.
- P2 Life Changing: this errors was seeded by the modelers. In the Doctor inform patient of results process action to move the CWP into the No longer needed state was removed by the modelers.
- P3 routine change: the modelers had included a Preferred mode of Communication gateway where the user is able to designate Mail, Auto-call, or MyHealthEvent. In the cases of Mail and MyHealthEvent, the tasks failed to move the CWP out of the Launched state. The error was corrected by removing the gateway as these options were hold-overs from a prior design that the designers had failed to remove.
- P4 no change: aside from the legacy gateway carried over from that prior design that was identical to the P3 routine change error, this workflow failed to move the CWP out of the Launched state. Neither the Make phone clinic appointment P4 nor the Discuss no change processes recorded the scheduled appointment, conversation, or resolution of the priority contact on any path. The error was corrected by modifying the tasks in Make phone clinic appointment P4 and Discuss no change. Tasks in these processes now change the state of the CWP indicating when the appointment is confirmed, the conversation takes place, and the doctor marks the contact as resolved.

The errors, though seemingly trivial, persisted through several reviews of the model, indicating the value of model checking if for nothing more than a sanity check on the connectivity of the model and the basic evolution of the CWP by the workflow.
Conclusion

The meaning of the model checking result is that the MATHflow model of the Priority Contact system implements the CWP under all possible inputs and under all possible ways for the actors to communicate. A faithful implementation of the MATHflow model would be correct relative to the intended work. The more general meaning is that a CWP can serve as a common entity which both clinicians and computers can change, thus allowing model checking technology to evaluate workflows that have both human-performed tasks and machine-preformed functions with vastly different capabilities in the same model. The MATHflow model and the CWP model could readily be translated into the input needed to apply established model checking tools. Extending model-checking to interactive health IT systems has great potential to advance HIT design and address safety in a powerful, formal method. To our knowledge no other approach can formally evaluate how well HIT is integrated with the fundamentally important manual activities of care. Before a system is actually built model checking can either prove that an interactive system can do the job it is supposed to do, or it identifies failures must be addressed.

Model checking proves a transition system implements a specification. To date, much of the research with model checking in health care has focused on formalizing workflows and proving temporal properties on workflows or on developing new modeling languages to describe workflows and human machine teaming in those workflows. The approach in this paper shifts focus to the work intended to be accomplished by a workflow. It uses model checking to prove that a workflow is capable to accomplish the work product that is specified in the CWP. Also, unlike prior work, this approach to model checking tries to use the CWP state diagram directly to prove the correctness of the system. This approach is in direct contrast to where the system specification is expressed in terms of formal logic.

Health IT is less mature than other established domains for system design. Absent a clear CWP specification and a technique for verifying at least partial algorithm correctness for producing a CWP, there is no basis to evaluate whether the workflow of a proposed HIT system is capable to accomplish the CWP. Further, it is not meaningful to compare systems on safety-related qualities such as usability if they have not been verified to produce equivalent work products. Consequently, HIT development may struggle with fundamental design decisions, such as how to achieve safe care improvements via appropriate allocation of functionality among clinical personnel and computing. Conversely, it is far more meaningful to compare workflows that have been verified as comparable for a variety of qualities such as reliability, usability, or efficiency, thus giving designers great latitude to find better solutions. Further, definitions of the checks could be instrumented as part of the HIT software to continue monitoring safety-related conditions after a verified system has been deployed. These capabilities should be important steps towards realizing the great potential of HIT.

Future Work

There are many directions opened up by the capability to verify socio-technical systems for healthcare. The new method first evaluates how the system will behave under idealized conditions, e.g., networks do not fail, and people show up for work on-time. We aim to investigate how the assumptions of the idealized system can then be relaxed to explore fault tolerance, and support designing to eliminate or mitigate the risks.

There are three critical pieces of future work related to the model checking step, itself: automation, scaling to large systems, results presentation, and workload analysis. The first step automates the process of extracting a Promela model from the MATHflow and CWP. It is critical for case studies on complex HIT systems. The second step characterizes the model checking on larger systems to understand its limits, and explores various techniques to mitigate state explosions including abstraction. The SPIN model checker typically scales to systems with millions of states, and it is hoped, but not yet known, that it is sufficient to verify interesting HIT processes. The third step relates directly to the usability of the model checking results. Any verification counter-example (i.e., a path that does not accomplish the work), must be mapped back into the MATHflow model and illustrated in MATHsim for the user to diagnose. We also plan to investigate how ontology modeling technology, with automated inferencing, can assist formally defining and scoping key CWP for clinical healthcare. These ontology models could provide a reusable inventory of key conceptual work products of clinical care to support health IT developers.

Acknowledgements

This research was supported by Grant No. 10510592 for Patient-Centered Cognitive Support under the Strategic Health IT Advanced Research Projects (SHARP) from the Office of the National Coordinator for Health Information Technology, and also by grant number R01HS021233 from the Agency for Healthcare Research and Quality. The content is solely the responsibility of the authors and does not necessarily represent the official views of the Agency for Healthcare Research and Quality.
References