Modes, Features, and State-Based Modeling for Clarity and Flexibility

Anitha Murugesan, Sanjai Rayadurgam, and Mats P. E. Heimdahl

Department of Computer Science and Engineering
University of Minnesota
200 Union Street, Minneapolis, Minnesota 55455, USA
{anitha, rsanjai, heimdahl}@cs.umn.edu

***Abstract*—The behavior of a complex system is frequently defined in terms of operational modes—mutually exclusive sets of the system behaviors. Within the operational modes, collections of features define the behavior of the system.**

Lucent and understandable modeling of operational modes and features using common state-based notations such as Statecharts or Stateflow can be challenging. In this paper we share some of our experiences from modeling modes and features in the medical device domain. We discuss the challenges and present a generic approach to structuring the modes and features of a generic Patient-Controlled Analgesia infusion pump.

***Index Terms*—Mode, Stateflow, Modeling Patterns**

I. Introduction

The dynamic behavior of a complex system is frequently defined in terms of *modes*, which are taken to be mutually exclusive sets of system behaviors [1]. The modes together with the rules defining when and how the system transitions between those modes are commonly referred to as the system's *mode logic* [2]. For many such systems, derivation and verification of the mode logic is challenging due to the plurality of modes and the complexity of the rules that govern the transitions. Often the problem is further exacerbated due to multiple orthogonal dimensions for partitioning the system behavior into modes, all of which may be necessary to adequately and succinctly describe the mode logic. Several well-known issues associated with mode logic have been explored; correct handling of complex mode transitions, ensuring consistency across orthogonal mode classifications, and eliminating mode confusion for system operators are some of the problems that have been studied in-depth [1], [3]. In addition, challenges arise when considering the essential book-keeping activities needed to manage various state variables, such as timers, which are invariably intertwined with the mode logic. Engineering considerations such as adaptability to change and reuse across multiple products in a product family further amplify this challenge. The challenges involved in modeling the modal behavior of systems have not, in our opinion, been adequately addressed in the literature. Even the definitions of frequently used terms—*modes*, *features* and *states*, to name a few—vary, are subjective, and often overlap, leading to some understandable confusion [4], [5].

This work has been partially supported by NSF grants CNS-0931931 and CNS-1035715.

When modeling the mode logic of a cyber-physical system in the medical device domain we had to squarely tackle this mode-modeling problem. In this short report we discuss several alternatives that were considered, the selected solution, and the rationale for our choice. We believe the experiences we gathered in the process are likely to be of benefit to others engaged in similar modeling efforts. More broadly, we hope our effort becomes a catalyst for the modeling research community to catalog solutions for various modeling problems and build a repertoire of *modeling patterns*.

II. Overview

The discrete behavior of complex control systems can be modeled in terms of (extended) finite state machines, which are formal, mathematical representations that support sophisticated verification techniques. As part of a larger project to investigate techniques for assuring safety and efficacy of medical cyber-physical systems, we modeled an infusion pump, called the Generic Patient Controlled Analgesia (GPCA) infusion system to better understand its behavior and to analyze its properties. Essential to this model was the mode logic of the infusion system. For our work we used the Simulink/Stateflow modeling tools [6] that provide a visual formalism for describing states and transitions in a modular and hierarchical fashion. The notations are also supported by a rich ecosystem of various analysis, translation, and verification tools. Since this modeling effort was carried out in the early stages of exploration of the system specification, one of the goals was to make it easy to effect changes to the model as our understanding of the system evolved. This, in particular, required a careful design of the mode logic model so that additions, modifications, and removal of behaviors could be done in a quick and localized fashion without losing model integrity. In the remainder of this report, we share our experiences and approaches used for modeling the mode logic of the GPCA.

To help the reader understand the examples and terminology used in the report, we begin by giving a brief overview of the GPCA mode logic in Section III. In Section IV, we briefly state some of the guiding principles behind our modeling approach. In Section V, we describe in some detail, the problems

encountered while attempting to model the mode logic of the GPCA and then delineate our approach for addressing those problems. We believe that this attempt is the first step towards a bigger goal of providing a catalogue of modeling patterns (Section VI).

III. GPCA Description

Infusion pumps are medical cyber physical systems used for controlled delivery of liquid drugs into a patient's body according to a physician's *prescription*, a set of instructions that governs the plan of care for that individual. Modern infusion pumps typically provide a variety of infusion options for drug delivery. In *basal* infusion, the drug is delivered at a constant (and usually low) rate for an extended period of time. In a *bolus* infusion, the drug is delivered at a higher rate for a short duration of time to address some immediate need or to increase the drug delivery according to some therapy regimen. There may be multiple bolus types. In *clinician bolus* infusion, the drug is delivered at an elevated rate in response to a clinician's request. For example, the clinician may prescribe an elevated rate of infusion for a period of time at the beginning of infusion therapy. An *intermittent bolus* infusion may be used to deliver additional drug at prescribed time intervals during the infusion therapy. Further, in a patient-controlled analgesia infusion system, a *patient bolus* infusion may be activated to deliver additional drug in response to a patient's request for more medication, typically to alleviate acute pain. More advanced pumps provide additional options such as *variable program* which allows varying the amounts, rates and times of delivery over the course of the infusion therapy, and *automatic ramping* which allows tapering the flow of drug at the beginning (and/or at the end) of the therapy.

With advances in technology these pumps have high-tech features, that allow the physicians to program therapies by choosing a combination of these options to support varying drug flow over time and based on patient requests. Such therapies involve switching and/or layering infusion types. In addition to providing an extensive range of options, these infusion pumps also have safety features designed to ensure that the device does not pose any hazard to the patient.

In this paper, we consider the software of a GPCA pump in which the programmable infusion types are basal, intermittent and patient bolus. The following sections describe the challenges in modeling such a variety of infusion types with an eye towards flexible and straightforward addition, modification and removal – it seems quite likely that clinical innovations will necessitate such changes to the mode-logic model of the GPCA software.

IV. Modeling Philosophy

We believe in a model-based approach for constructing reactive systems, in which the model plays a central role throughout the life-cycle by serving as a concrete representation of the conceptual links between the various artifacts produced during development. At its essence, modeling is a design activity and adhering to basic design principles while modeling has several benefits. Models must be conceptually clean, easy to understand and maintain, and be amenable to formal analysis. Maintainability requires that the constructed model should be easy to modify for extensions and contractions [7]. In the present instance, our immediate goal of the modeling exercise was to explore and enhance our understanding of the requirements by identifying interesting interactions of various system behaviors that may have to be clarified with experts in the domain.

V. Modeling Concerns

In the sequel we use the terms *modes* and *features*. Features are visible aspects of the system from a user's perspective that can be individually enabled [8]. A mode is an exclusive collection of system behaviors that is exhibited [1] from the system's perspective. While the definition of these terms and the distinction between them are of considerable interest and subject to debate, we avoid such discussions here. For the purposes of this paper, we refer to the available infusion types of the GPCA as features. For example, basal, patient bolus and intermittent infusions are features that are available in the device and the physician can choose and program a prescription that specifies the necessary parameters for these features. The resulting system behavior is modal – at any given instant there is a specific infusion type being delivered, such as basal or patient bolus. The mode logic defines how the enabled features interact to produce the system behavior for each mode.

In representing the mode logic of the GPCA as finite state machines, there are three main concerns that have to be effectively addressed. A simplistic view of different system modes as top-level states with transitions to represent mode changes, though aligning with a user's view of the system, lead to inflexible designs that hinder changes to the mode logic. Further, the book-keeping logic of internal state variables such as timers and condition flags are inextricably linked to the states and transitions that complicates the task of effecting changes as our system understanding evolves. Finally, the behaviors of the individual modes have both variety and similarity which are neither captured nor exploited in a flat finite state model of mode logic.

A. Modeling GPCA Mode Logic

During requirements analysis, the various delivery modes were visualized from the user's perspective as shown in Figure 1a. In our initial attempt to model these modes, we arranged the modes as sequential states with their respective timers and flags contained within them. Although it appeared superficially intuitive and corresponded well with our mental models, complications emerged as we attempted to affect changes to the mode logic. For example, Figure 1b visually shows the complexity involved while adding new modes to the existing mode logic. While a highly competent engineer may meticulously handle changes while retaining model integrity, it became evident that the design is not easily adaptable for evolution and maintenance.

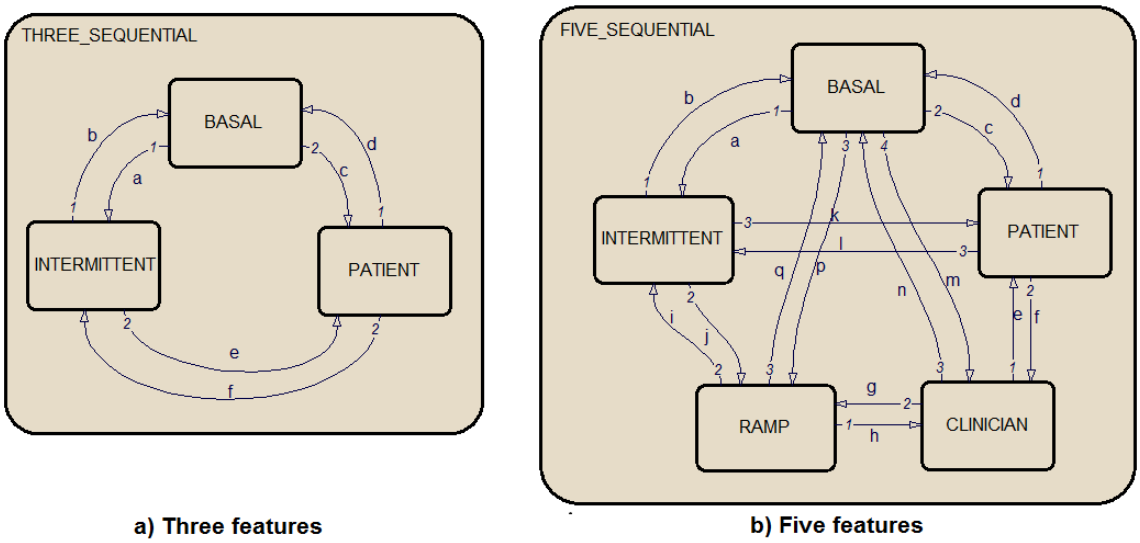

Fig. 1. Sequential Structure of Mode Logic

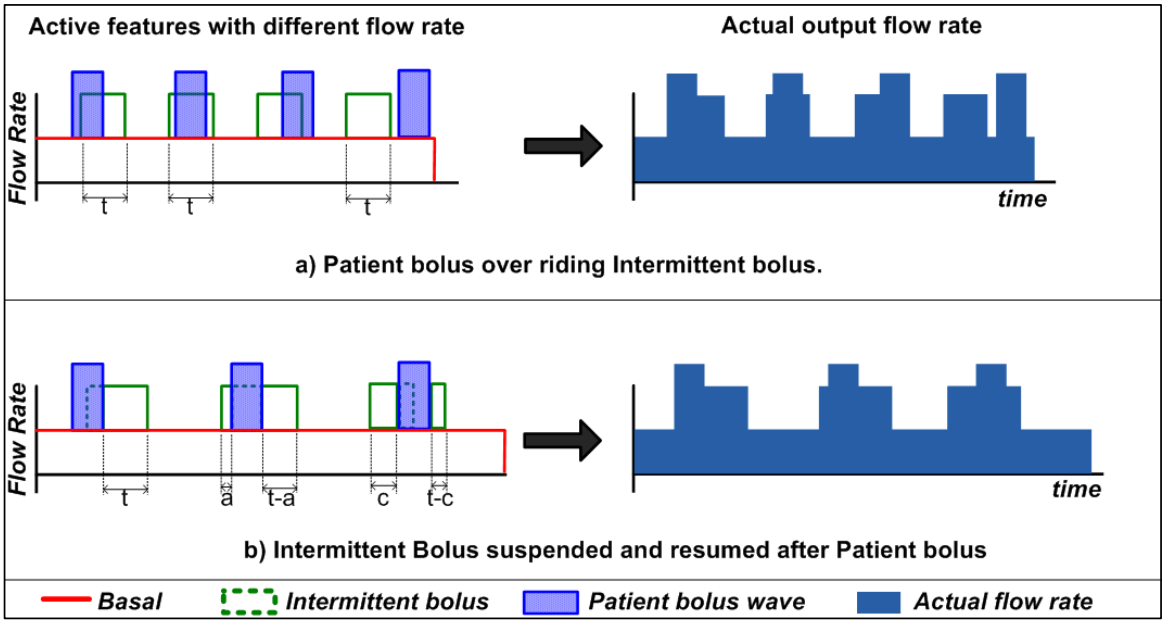

Fig. 2. Variety of System Behaviors

Another concern was flexibility in the design for accommodating changes in behavioral requirements. For example, a version of a GPCA software requirement reads: *When an intermittent bolus is in progress and a patient bolus is requested, then the intermittent bolus is suspended until the patient bolus is delivered and resumed after the patient bolus.* This may be taken to mean that the total duration of time from the beginning of the intermittent bolus to its end is the sum of the intermittent and patient bolus delivery times as as illustrated in Figure 2b. Such a behavior may be in conflict with a different requirement that determines the total duration of infusion therapy. Alternatively, it may also be reasonable to expect that the total duration is simply the programmed duration of the intermittent bolus as shown in Figure 2a, since patient bolus is typically a high dosage non-periodic infusion for a very short time period. If the ongoing infusion was basal instead of intermittent bolus, this indeed seems to be the expected behavior.

These alternatives have different consequences not only from the domain perspective but also from a modeling perspective. The mode logic for a *suspend-resume-bolus* behavior is different from the mode logic for a *highest-rate-bolus* behavior. During requirements analysis we want to be able to easily change the the mode logic to simulate the alternatives and let the domain experts make the appropriate choice. As designers, we foresee that such behavioral requirements are

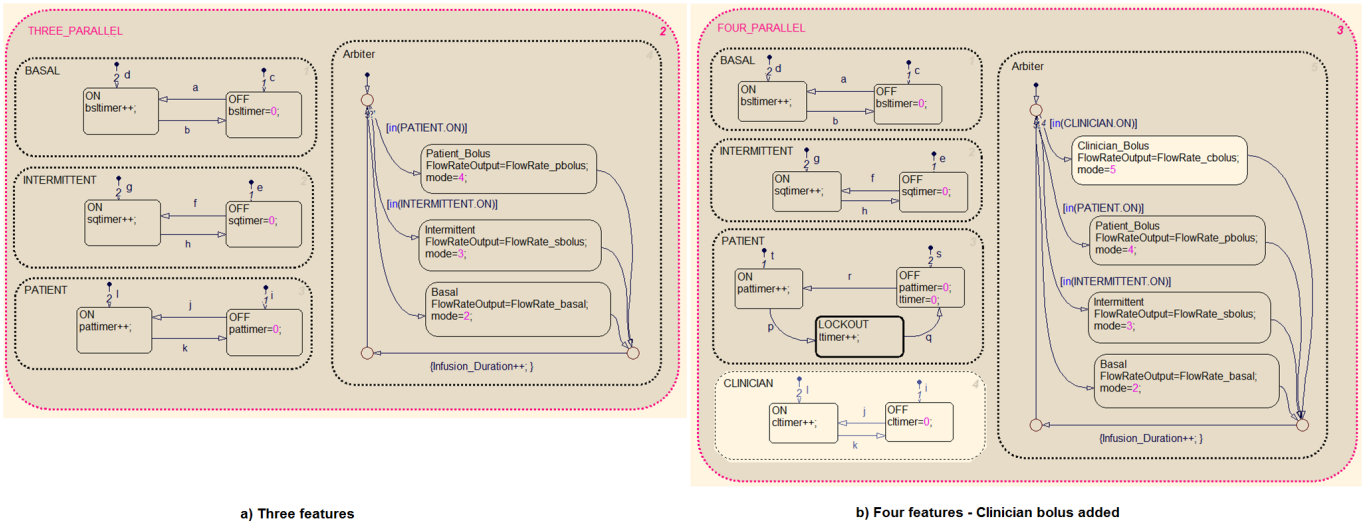

Fig. 3. Parallel Structure of Mode Logic

likely to change and that there may be opportunities for reusing ”generic” descriptions of modal behavior. Accomplishing this with a sequential state machine for the mode-logic is cumbersome because the associated book-keeping and timer management tends to be dispersed across multiple states and transitions.

An alternative approach is to design the system’s modal behavior as a composition of multiple *feature* behaviors mediated and resolved by an *arbiter* as shown in Figure 3a. In this structure, each feature is a self contained state machine that has its own logic for turning on and off. The individual timers for each feature are managed within its state machine and the overall system timer for therapy duration is commonly handled independently of the individual state machines. The transitions within each state machine are independent of the other parallel machines. The arbiter is a separate parallel state machine that decides the modal behavior based on the individual feature behaviors. The logic of arbitration between the feature behaviors to decide the system model behavior, that resides in the arbiter, varies with the domain and application. In the case of the GPCA, the arbiter handles the prioritization and feature interactions to determine the overall system behavior, that is manifested by the flow rate used for infusion and current mode displayed to the user.

We believe that this pattern results in a design that is modular, scalable and easier to understand. For example, in the GPCA, all the logic for prioritization was grouped in the arbiter and the individual timer and transition logic is within the respective feature state machine itself. This makes it straightforward to incorporate changes or extend the model to include new features. As an example of the extensibility afforded by this design, the infusion system with a new bolus feature (CLINICIAN bolus), illustrated in Figure 3b, has been included with minimal change that was made possible by the inherent modularity. The observant reader would have noticed a change in the PATIENT bolus feature between the Figures 3a and 3b - the addition of LOCKOUT state - that illustrates the flexibility in the design to accommodate the changes in system behavioral requirements. Similarly, one can easily modify the model to effect a suspend-resume behavior for the INTERMITTENT feature,as illustrated in Figure 2b, by adding a new state along with transitions for suspend and resume within the INTERMITTENT state machine and corresponding logic in the arbiter for the prioritization. The parallel structuring allows the design to be modular by aggregating the prioritization logic in the arbiter and the feature specific behaviors within the respective state machines, without affecting the others.

B. Complex Features

In the GPCA, some bolus modes have additional constraints placed on their occurrence. For example, once a patient bolus infusion is completed, the feature should be locked out from further activation (temporarily disabled) for a preset amount of time so that the system cannot continually provide high dosage infusion. This is easily accommodated by having an additional LOCKOUT state and appropriate transitions to and from that state, within the PATIENT bolus state machine. This may also viewed as a sub-state of the OFF state. Similarly, a feature may have to be temporarily suspended and then resumed at a later point; this can be seen as a refinement of the ON state into sub-states. Instances of such variety in feature behaviors and comparable approaches for modeling such behaviors can be found in other safety-critical domains such as flight guidance systems [9].

Analyzing such common behavioral patterns of features lead us to expand the simple on-off feature state-machine to a more advanced hierarchical machine pattern as shown in Figure 4. Not all states in this pattern may be required for a single feature and, indeed, each GPCA feature was modeled with only a few of these states. However, we believe that this general pattern captures the common structure of such

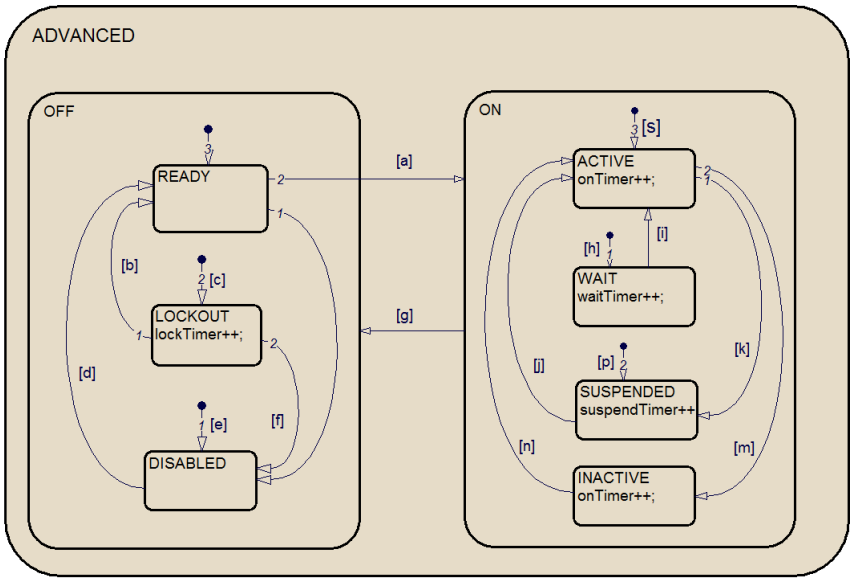

Fig. 4. Complex Features

feature state machines that typically arise when modeling mode-logic. In our limited experience with the GPCA system, we found that tailoring the common pattern to suit the needs for modeling a specific feature was rather straightforward. It helps analyzing the mode logic and effecting changes to it without losing model integrity, as we added new features and made changes to existing ones. Additionally, this common pattern helps handling the book keeping of timers and other state variables in a consistent fashion across modes.

The parallel structure of the mode logic allowed inclusion of this complex feature state machine without affecting the way the other features are modeled. This easy inclusion/exclusion of features states was one of the design goals of the model.

C. Summary

In our endeavor to model the mode logic for GPCA software, we identified interesting and challenging issues in terms of design flexibility. We also identified some commonality in the behavior within features. While the sequential structure better matches the user's conceptual view, it was clearly not suitable from a design perspective. An alternative approach using state transition tables for describing the mode logic becomes unmanageable with increase in number of features or parameters that influence the mode logic. We believe that the modeling approach in this paper, with some generalization, could be applied as a pattern for representing the mode-logic of reactive systems in several domains.

VI. Conclusion

In this paper, we discussed our approach for modeling the mode logic of a medical device and the rationale for our design choices. The challenges that had to be addressed and the modeling technique we employed seem to be common to several domains such as avionics, automotive controls and medical devices. We believe that identifying, documenting and sharing such useful solutions is of value and we hope this grows into a broader initiative towards a comprehensive catalog of modeling patterns for cyber-physical systems.

Acknowledgments

We would like to acknowledge Dr. Steven P. Miller of Rockwell Collins' Advanced Technology Center for starting us thinking about modeling modes a decade ago and Dr. Michael W. Whalen of the University of Minnesota for many discussions related to the structure of state-based models.

References

[1] N. Leveson, L. D. Pinnel, S. D. Sandys, S. Koga, and J. D. Reese, "Analyzing software specifications for mode confusion potential," in *Proceedings of a Workshop on Human Error and System Development*, 1997, pp. 132–146.

[2] A. Joshi, S. P. Miller, and M. P. Heimdahl, "Mode confusion analysis of a flight guidance system using formal methods," in *In Proceedings of 22nd Digital Avionics Systems Conference(DASC'03)*, vol. 1. IEEE, 2003, pp. 2–D.

[3] S. P. Miller, J. N. Potts, and R. Collins, *Detecting mode confusion through formal modeling and analysis*. National Aeronautics and Space Administration, Langley Research Center, 1999.

[4] J. Bredereke and A. Lankenau, "A rigorous view of mode confusion," *Computer Safety, Reliability and Security*, pp. 1–13, 2002.

[5] E. Pulvermueller, A. Speck, J. Coplien, M. DHondt, and W. De Meuter, "Feature interaction in composed systems," *Object-Oriented Technology*, pp. 1–16, 2002.

[6] Stateflow - environment for modeling state machines. [Online]. Available: http://www.mathworks.com/products/stateflow/

[7] D. Parnas, "Designing software for ease of extension and contraction," *IEEE Transactions on Software Engineering*, no. 2, pp. 128–138, 1979.

[8] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," DTIC Document, Tech. Rep., 1990.

[9] S. P. Miller, A. Tribble, T. Carlson, and E. J. Danielson, *Flight guidance system requirements specification*. National Aeronautics and Space Administration, Langley Research Center, 2003.