Modeling and Integrating Physical Environment Assumptions in Medical Cyber-Physical System Design

Zhicheng Fu, Chunhui Guo, Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
{zfu11, cguo13}@hawk.iit.edu, ren@iit.edu

Yu Jiang
School of Software
Tsinghua University
Beijing, China
jy1989@mail.tsinghua.edu.cn

Lui Sha
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
lrs@illinois.edu

***Abstract*—For a cyber-physical system, its execution behaviors are often impacted by its operating physical environment. However, the assumptions about a cyber-physical system's expected physical environment are often informally documented, or even left implicit and unspecified in system design. Unfortunately, such implicit physical environment assumptions made by safety critical cyber-physical systems, such as medical cyber-physical systems (M-CPS), can lead to catastrophes. Several recent U.S. Food and Drug Administration (FDA) medical device recalls are due to implicit physical environment assumptions. In this paper, we develop a mathematical assumption model and composition rules that allow M-CPS engineers to explicitly and precisely specify assumptions about the physical environment in which the designed M-CPS operates. Algorithms are developed to integrate the mathematical assumption model with system model so that the safety of the system can be not only validated by both medical and engineering professionals but also formally verified by existing formal verification tools. We use a FDA recalled medical ventilator scenario as a case study to show how the mathematical assumption model and its integration in M-CPS design may improve the safety of the ventilator and M-CPS in general.**

I. Introduction

For a cyber-physical system, its execution behaviors are often impacted by its operating physical environment. However, the assumptions about a cyber-physical system's expected physical environment are often informally documented, or even left implicit and unspecified in system design [1]. Unfortunately, implicit physical environment assumptions made by safety critical cyber-physical systems, such as medical cyber-physical systems (M-CPS), can lead to catastrophes. We use two recent U.S. Food and Drug Administration (FDA) medical device recalls [2] to illustrate how implicit physical environment assumptions have caused M-CPS failures.

FDA Medical Device Recall 1. ***Dräger Medical, Evita V500 and Babylog VN500 Ventilators — Faulty Batteries, July 13, 2015 [3].*** *FDA has identified this as a Class I recall, the most serious type of recall. The battery capacity of optional PS500 Power Supply Unit of the Infinity ACS Workstation Critical Care (Evita Infinity V500) did not last as long as expected. The batteries installed in the PS500 depleted much earlier than expected although the battery indicator showed a sufficiently charged battery. Even when the battery depleted totally, the power fail alarm was not generated. If the ventilator shuts down without alarm, a patient may not receive necessary oxygen. This could cause patient injury or death.*

In the FDA recalled ventilator, there are three major components in the system: *Controller*, *Alarm* and *Battery* [4]. The *Controller* calculates remaining time that the *Battery* can supply. If the remaining time is within the range of 30 to 35 minutes, the *Controller* will send an event to the *Alarm* component to trigger an alarm for medical staffs. One implicit assumption is that ventilators are installed in temperature controlled areas, such as hospital rooms, where room temperature are maintained at normal room temperature. In this environment, the capacity the battery can supply is a constant. However if hospital rooms are unable to maintain the assumed operating temperature, the capacity of the battery is reduced. This unanticipated change of battery capacity will cause *Controller* to miscalculate the remaining time and hence fail to send an alarm event before the ventilator is out of power.

FDA Medical Device Recall 2. ***HeartWare Ventricular Assist System (VAS)–Electrostatic Discharge May Cause Pump Failure, February 2, 2015 [5].*** *A buildup of static may cause a sudden discharge of electricity (electrostatic discharge) in the device. When this happens, data in the motor controller that manages the pump's operation may be corrupted and the device may stop working.*

In this recall, one implicit physical environment assumption is that the ambient humidity of this device is be maintained in a normal range. If the device operates in an environment which has lower humidity than implicitly assumed, it can trigger buildup of static electricity resulting in adverse health consequence, including death.

These recalls and many other examples that can be found in FDA recall database [2] show an inarguable fact that implicit assumptions about M-CPS's physical environment are dangerous and can lead to loss of human life. Hence, being able to explicitly and accurately specify physical environment assumptions and integrate these assumptions in M-CPS design and development is critical to ensure the safety of M-CPSs.

The design and development of M-CPS requires knowledge from both engineering and medical fields and collaboration between engineers, computer scientists, and medical profession-

als. Engineers are more familiar with mathematical structures and operations whereas medical professionals are more used to statecharts as disease and treatment models often have high resemblance to statecharts. Hence, to explicitly take physical environment assumptions into M-CPS design, our strategy is to define a mathematical model and composition rules for engineers to explicitly and accurately specify physical environment assumptions. The mathematical assumption model is then automatically transformed into a statechart model and integrated with system statechart models so that the integrated models can be validated by both medical and engineering professionals and system safety properties can be formally verified with existing model verification tools. Fig. 1 depicts the high level view of our M-CPS design architecture.

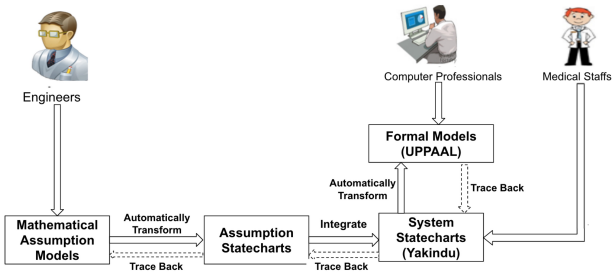


Fig. 1. M-CPS Design Architecture with Physical Environment Assumptions

The rest of the paper is organized as following: we discuss related work about assumption management in Section II. Section III defines the mathematical assumption model, its structure and composition rules. Section IV discuses the transformation and integration of the mathematical assumption model into M-CPS design. Section V uses one FDA recall example as a case study to show how M-CPS safety can be improved by integrating physical environment assumptions into system design. We draw conclusions and point out future work in Section VI.

II. Related Work

Our work is motivated by the guidance "Applying Human Factors and Usability Engineering to Medical Devices" released by U.S. Food and Drug Administration [6]. In the guidance, it recommends developers to evaluate and understand relevant characteristics of all intended use environments and describe them for the purpose of safety evaluation and design [6]. Our work is also motivated by the ongoing efforts of the Medical Device Plug-and-Play Interoperability (MDPnP) program [7]. This program has been leading the development of the Integrated Clinical Environment (ICE) standard and gap analysis on the ability of the IEEE 11073 family of standards [8] to meet the clinical use cases described in the ICE standard. Currently, much work about assumptions for MDPnP focuses on establishing dynamic connectivity of devices with different data format assumptions [9], synchronization among devices with diverse clock assumptions [10], and ensuring fair access to a communication medium [11]. Compliment to the MDPnP's efforts in addressing system assumptions issues, our work focuses on how to prevent system failures due to implicit physical environment assumptions.

Implicit assumptions are a main factor that are determined to be the cause of failures in safety critical cyber-physical systems [12]. This is experienced in several industry and academic projects. The Ariane 5 [12] and Child-seat Airbag Incident [13] are the results of implicit assumptions made in system design. The problem are magnified further in M-CPS development such as the notorious incidents in the 80's involving the Therac 25 radiation therapy machines [14], and many recent medical device recalls documented in FDA recall database [2]. Much work has been done for explication of assumptions in system design [15], [16]. An assumption management framework has been introduced by Tirumala with the aim of designing a set of well-defined vocabularies to encode architectural assumptions of a system [12]. Instead of explicating architectural assumptions as these previous work, in this paper, we try to explicate physical environment assumptions and integrate these assumptions in M-CPS design to improve safety.

Many efforts have been also done in applying formal methods to medical device analysis for improving safety [17], [18], [19]. The use of formal methods has started to influence actual medical device review procedures [20]. However, much of the formal analysis work has been done without considering the influence of unpredictable physical environment conditions on safety of medical devices. We intend to take the initiative to move forward in this direction.

III. Physical Environment Assumption Model

A. Mathematical Structure of Physical Environment Assumptions

The impact of physical environment change on M-CPS' behaviors is often due to the fact that physical environment change can cause M-CPS system parameters to change, as in the two FDA recalls presented in Section I. Hence, to bring to light physical environment assumptions in M-CPS, we need to make the dependences between system parameters and environment parameters. We use a 2-tuple $(\texttt{name}, \texttt{type})$ to represent both system parameters (s_i) and environment parameters (e_j). One environment parameter, such as temperature, can impact multiple system parameter values; and a single system parameter can be influenced by multiple environment variables. For a given set of system parameters $S = \{s_1, \ldots, s_i, \ldots, s_n\}$ and physical environment parameters $E = \{e_1, \ldots, e_j, \ldots, e_m\}$, we define the following mathematical structure for specifying physical environment assumptions.

Definition 1 (Physical Environment Assumption)**.** *A physical environment assumption $A(s, E', D)$ is defined as the dependences (D) between a system parameter s and a set of environment parameters in E', where $s \in S$, $E' \subseteq E$, and $D = \{d_1, \ldots, d_k, \ldots, d_l\}$ is the set of dependences.*

Definition 2 (Dependence)**.** *A dependence d in an assumption A is represented by a 3-tuple $(V(s), C(E'), p(C(E')))$, where $C(E')$, $V(s)$, and $p(C(E'))$ are the* `condition(s)` *of environment parameter(s), system parameter* `value` *and the impact level of the environment parameters on the system parameter s under the given environment* `condition(s)`*, respectively.*

In the assumption structure, the impact level $p(C(E'))$ is an integer indicating the significance of a given set of environment conditions on a specific system parameter, the smaller the value, the higher the impact. For instance, both temperature and humidity can impact the value of battery's capacity, but the significance of the impact may be different.

In an assumption $A(s, E', \{d_1, \ldots, d_k, \ldots, d_l\})$, the environment condition $C(E')$ of all dependences D are mutually exclusive, i.e., at any given time, only one environment condition $C_k(E')$ holds. All physical environment assumptions in a M-CPS are represented by $\mathcal{A} = \{A_1, \ldots, A_n\}$. We use the following example to illustrate the use of these defined mathematical structure in representing physical environment assumptions in a M-CPS.

Example 1. *Consider the following two scenarios of battery behaviors in the FDA Recall 1 example. Assume the initial battery capacity is 1Ah, meaning that the battery provides 1A for one hour [21].*

- ***S1**: when the temperature T is within the range $[15°C, 35°C]$, the battery capacity does not change, i.e., $C = 1$ [21];*
- ***S2**: when the temperature T is within the range $[-10°C, 15°C)$, the battery capacity is $C = 1 - 2 \times (25 - T)/100$ [21].*

*The system only contains one system parameter, i.e., battery capacity $c = (C, \texttt{real})$, and one environment parameter, i.e., temperature $t = (T, \texttt{real})$. Assume different temperatures impact the battery capacity at the same level 1, the two scenarios **S1** and **S2** indicate that the physical environment assumption about battery capacity and temperature contains two dependences, i.e., $A_1(c, \{t\}, D_1)$, where $D_1 = \{d_1, d_2\}$, $d_1 = (1, 15 \leq T \leq 35, 1)$, and $d_2 = (1 - 2 \times (25 - T)/100, -10 \leq T < 15, 1)$.*

Besides temperature, the battery capacity can also be affected by humidity in the following two scenarios.

- ***S3**: when the relative humidity H is in range $[10\%RH, 30\%RH]$, the battery capacity reduces by 10%, i.e., $C = 0.9$ [22];*
- ***S4**: when the relative humidity H is in range $[40\%RH, 60\%RH]$, the battery capacity does not change, i.e., $C = 1$ [22].*

Assume the humidity's impact on battery capacity is more significant when $0.1 \leq H \leq 0.3$. The assumption about the impact of humidity on battery capacity can be represented by $A_2 = (c, \{h\}, D_2)$, where $h = (H, \texttt{real})$, $D_2 = \{d_3, d_4\}$, $d_3 = (0.9, 0.1 \leq H \leq 0.3, 1)$, and $d_4 = (1, 0.4 \leq H \leq 0.6, 2)$.

In a M-CPS system, multiple environment parameters can impact the same system parameter. For instance, both temperature and humidity can impact battery capacity. In this case, we can represent the environment assumptions as two separate assumptions with single environment parameter as A_1 and A_2 in Example 1 or one assumption with multiple environment parameters $A_3(c, \{t, h\}, D_3)$. From engineers' perspective, multiple assumptions with single environment parameter is more intuitive and easy to validate, while from model integration and formal verification perspective, single assumption with multiple environment parameters dependency avoids interferences among different assumptions on the same system parameter. Hence, to facilitate integrating assumptions into system design, next we discuss assumption compositions under the defined assumption model.

B. Assumption Composition

Given two assumptions $A_1(s_1, E_1', D_1)$ and $A_2(s_2, E_2', D_2)$, if $s_1 = s_2$, we call these two assumptions *interfering* assumptions. We define the composition operation ($\oplus$) to compose two *interfering* assumptions $A(s, E', D) = A_1(s, E_1', D_1) \oplus A_2(s, E_2', D_2)$ as following:

- **Composition Rule 1**: $E' = E_1' \cup E_2'$
- **Composition Rule 2**: $D = D_1 \otimes D_2$, where the operation $\otimes$ is defined by the followed rules;
- **Composition Rule 3**: Set the initial value of D as $\emptyset$, for each dependence $d_i(V(s), C(E_1'), p(C(E_1'))) \in D_1$ and each dependence $d_j(V(s), C(E_2'), p(C(E_2'))) \in D_2$, recursively perform $D = D \cup d(V(s), C(E'), p(C(E')))$, where
  - **Composition Rule 3.1**: $C(E') = C(E_1') \wedge C(E_2')$
  - **Composition Rule 3.2**: $p(C(E')) = \min\{p(C(E_1')), p(C(E_2'))\}$
  - **Composition Rule 3.3**: If $p(C(E_1')) < p(C(E_2'))$, $V(s)|d = V(s)|d_i$; if $p(C(E_1')) > p(C(E_2'))$, $V(s)|d = V(s)|d_j$; otherwise, $V(s)|d$ is set as the worst-case among $V(s)|d_i$ and $V(s)|d_j$.

The operation (|) in $V(s)|d$ obtains the system parameter's value $V(s)$ in the corresponding dependence d. Noting that the worst case of a system parameter's value is defined by domain experts. For instance, in Recall 1, the worst case of the battery capacity is the lowest value. We use Example 2 to show the composition process of *interfering* assumptions.

Example 2. *As both A_1 and A_2 given in Example 1 are related to battery capacity c, they are interfering assumptions. We perform the composition $A_3(c, E_3', D_3) = A_1(c, \{t\}, D_1) \oplus A_2(c, \{h\}, D_2)$ as follows. Applying **Composition Rule 1** and **Composition Rule 2**, $E_3' = \{t, h\}$ and $D_3 = \{d_{13}, d_{14}, d_{23}, d_{24}\}$. We take d_{13} as an example to show how to apply **Composition Rule 3** to compose d_1 and d_3. According to **Composition Rule 3.1** and **Composition Rule 3.2**, $C(E_3')|d_{13} = 15 \leq T \leq 35 \wedge 0.1 \leq H \leq 0.3$ and $p(C(E_3'))|d_{13} = 1$. As the impact level in d_1 and d_3 are both equal to 1, based on **Composition Rule 3.3**, $V(c)|d_{13}$ is set as the worst-case of $V(c)|d_1$ and $V(c)|d_3$, i.e., the smaller battery capacity value 0.9. Similarly, we apply the rules to other three dependences. The composed dependencies are:*

$$d_{13} = (0.9, 15 \leq T \leq 35 \wedge 0.1 \leq H \leq 0.3, 1)$$
$$d_{14} = (1, 15 \leq T \leq 35 \wedge 0.4 \leq H \leq 0.6, 1)$$
$$d_{23} = (1 - \frac{2(25 - T)}{100}, -10 \leq T < 15 \wedge 0.1 \leq H \leq 0.3, 1)$$
$$d_{24} = (1 - \frac{2(25 - T)}{100}, -10 \leq T < 15 \wedge 0.4 \leq H \leq 0.6, 1)$$

If a system contains more than two assumptions, we apply the composition operation iteratively until all assumptions are non-*interfering* , i.e, related to different system parameters.

According to **Composition Rule 1-3**, we derive Algorithm 1 and Algorithm 2 to perform the composition operation and the iterative composition process, respectively.

Algorithm 1 TWO-COMPOSE(A_1, A_2)

Input: Two *interfering* assumptions $A_1(s, E_1', D_1)$ and $A_2(s, E_2', D_2)$
Output: The composed assumption $A(s, E', D)$

$E' = E_1' \cup E_2'$, $D = \emptyset$
for each $d_i(V(s), C(E_1'), p(C(E_1'))) \in D_1$ **do**
  for each $d_j(V(s), C(E_2'), p(C(E_2'))) \in D_2$ **do**
    Create a new dependence $d(V(s), C(E'), p(C(E')))$
    $C(E') = C(E_1') \wedge C(E_2')$
    if $p(C(E_1')) < p(C(E_2'))$ **then**
      $p(C(E')) = p(C(E_1'))$, $V(s)|d = V(s)|d_i$
    else if $p(C(E_1')) > p(C(E_2'))$ **then**
      $p(C(E')) = p(C(E_2'))$, $V(s)|d = V(s)|d_j$
    else
      $p(C(E')) = p(C(E_1'))$
      $V(s)|d$ is set as worst case of $V(s)|d_i$ and $V(s)|d_j$
    end if
    $D = D \cup d$
  end for
end for
return A

Algorithm 2 ASSUMPTION COMPOSITION

Input: A system's assumption set $\mathcal{A}$
Output: The composed assumption set $\mathcal{A}'$

Divide $\mathcal{A}$ into subsets $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ based on different system parameters
for each $\mathcal{A}_i \in \mathcal{A}$ **do**
  Create a new assumption $A = A_1|\mathcal{A}_i$
  $\mathcal{A}_i = \mathcal{A}_i \setminus A_1|\mathcal{A}_i$
  while $\mathcal{A}_i \neq \emptyset$ **do**
    $A =$TWO-COMPOSE$(A, A_1|\mathcal{A}_i)$
    $\mathcal{A}_i = \mathcal{A}_i \setminus A_1|\mathcal{A}_i$
  end while
  $\mathcal{A}' = \mathcal{A}' \cup A$
end for
return $\mathcal{A}'$

IV. INTEGRATING PHYSICAL ENVIRONMENT ASSUMPTION MODELS WITH SYSTEM MODEL

To validate and verify the correctness of assumption models, we need to integrate assumption models with system models. In M-CPS domain, it is important that engineers and medical staffs can understand M-CPS models easily and validate them through user-friendly simulation. Noting that statechart has high remembrance to disease and treatment models, can be easily understood by field professionals, is executable, and can be indirectly verified, we hence transform the mathematical model of physical environment assumptions to statecharts. We choose Yakindu statechart tool. Yakindu is an open-source tool kit based on the concept of statecharts. It has a well-designed user interface, provides simulation and code generation functionalities, and hence enables rapid prototyping and validation with field professionals.

A. Transforming Assumption Models to Statecharts

The interferences among multiple assumptions on the same system parameter increase the difficulty of transforming mathematical assumption models into statecharts. We first use Algorithm 2 to compose assumptions and remove their interferences. For the transformation purpose, we can then assume all assumptions $A \in \mathcal{A}$ are non-*interfering*. For each non-*interfering* assumption $A \in \mathcal{A}$, we use an independent sub-statechart in an orthogonal state to represent the assumption. For each dependence $d(V(s), C(E'), p(C(E')))$ in the assumption A, we create a state $\mathtt{S_d}$ with entry action $s = V(s)$ and add a transition from the initial state to state $\mathtt{S_d}$ with guard $C(E')$ to represent the system parameter's corresponding change under the environment condition $C(E')$. The added transition's priority is set to be $p(C(E'))$. For all states in the sub-statechart except the initial state, we add transitions back to the initial state with guard `true` to enable the statechart capturing environment conditions. Algorithm 3 depicts the transform procedure. Example 3 illustrates how we apply Algorithm 3 to transform A_3 in Example 2 to a statechart.

Algorithm 3 TRANSFORM ASSUMPTIONS TO STATECHARTS

Input: A system's assumption set $\mathcal{A}$.

1: Create a statechart `Environment`
2: Add a orthogonal state `Assumptions` to `Environment`
3: **for** each $A_i(s_i, E_i', D_i) \in \mathcal{A}$ **do**
4: Add a sub-statechart $\mathtt{st_i}$ to `Assumptions`
5: Add the initial state `InitState` to $\mathtt{st_i}$
6: **for** each $d_j(V(s_i), C(E_i'), p(C(E_i')) \in D_i$ **do**
7: Add a state $\mathtt{sd_j}$ to $\mathtt{st_i}$ with entry action $s_i = V(s_i)|d_j$
8: Add a transition $\mathtt{T_j}$ from `InitState` to $\mathtt{sd_j}$ with guard $C(E_i')$
9: Add a transition from $\mathtt{sd_j}$ to `InitState` with guard `true`
10: Set the priority of $\mathtt{T_j}$ as $p(C(E_i'))$
11: **end for**
12: **end for**

Example 3. *To transform A_3 in Example 2, we create the statechart* `Environment` *that contains an orthogonal state* `Assumptions`*. To represent A_3, we add a sub-statechart* `batteryassumptions` *containing an initial state* `Init_State` *to the orthogonal state* `Assumptions`*. For $d_{13} \in A_3$, we create state* $\mathtt{d_{13}}$ *with entry action $c = 0.9$, add a transition from state* `Init_State` *to* $\mathtt{d_{13}}$ *with guard $T >= 15$ && $T <= 35$ && $H >= 0.1$ && $H <= 0.3$, and set the priority of the added transition as 1. According to the Line 9 in Algorithm 3, we also add a transition from state* $\mathtt{d_{13}}$ *back to* `Init_State` *with guard* `true`*. Similar procedures can be taken for the rest dependences in A_3.*

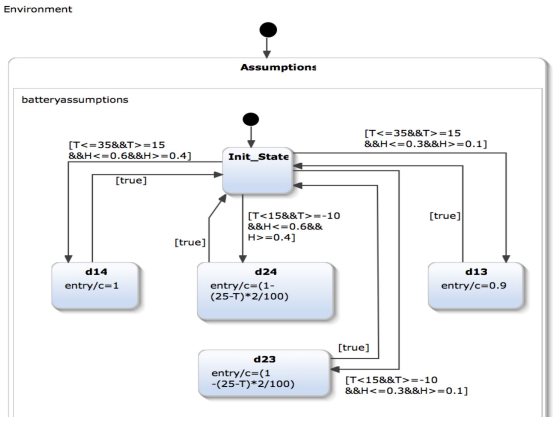


Fig. 2. Assumption Statechart

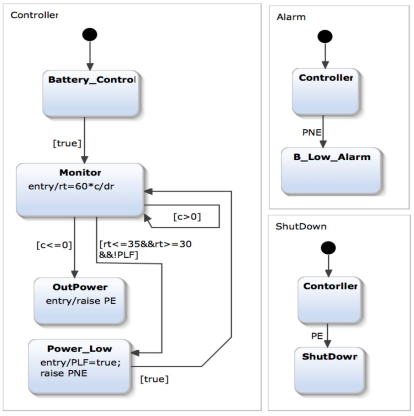


Fig. 3. Medical Ventilator Model without Environment Assumptions

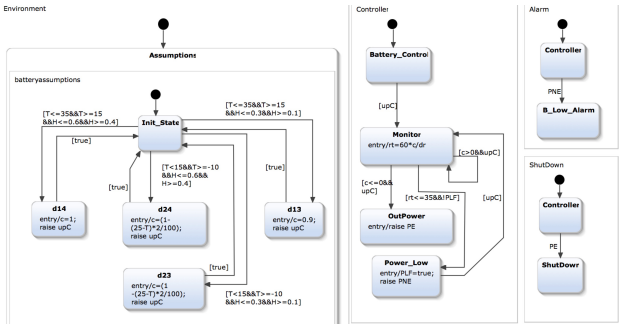


Fig. 4. Medical Ventilator Model with Environment Assumptions

B. Integrating Assumption Statecharts with System Model

To integrate assumption statecharts with system model, we model the interactions between assumption statechart models and system statechart models with following rules:

- **Integration Rule 1**: For each system parameter s, declare an event e_s to implement the interaction;
- **Integration Rule 2**: For each state in the assumption statechart model, if it changes the value of a system parameter s, raise the event e_s in the state's entry action;
- **Integration Rule 3**: For the system statecharts, modify it by the followed rules;
  - **Integration Rule 3.1**: For each transition $T(G, A)$, if its guard G or action A involves a system parameter s, $G = G \;\&\&\; e_s$;
  - **Integration Rule 3.2**: For each state, if its action involves a system parameter s, replace the guard of all its incoming transitions $\{T_i(G_i, A_i)\}$ by $G_i = G_i \;\&\&\; e_s$.

We integrate the assumption statechart shown in Fig. 2 with the medical ventilator statechart model in Fig. 3. and Fig. 4 shows the integrated system statechart model. In particular, based on **Integration Rule 1**, we declare an event `upC` for the battery capacity c. According to **Integration Rule 2**, we raise the event `upC` in the entry action of all states in the sub-statechart `batteryassumptions` except state `Init_State`. In the ventilator model, there are two transitions involving the battery capacity c: transition $T_1(G_1, A_1)$ from state `Monitor` to state `OutPower` and the self-loop transition $T_2(G_2, A_2)$ of state `Monitor`, where $G_1 = [c <= 0]$ and $G_2 = [c > 0]$. Based on **Integration Rule 3.1**, the two transitions' guards are set as $G_1 = [c <= 0 \;\&\&\; \texttt{upC}]$ and $G_2 = [c > 0 \;\&\&\; \texttt{upC}]$. The only one state involving battery capacity c in the ventilator model is `Monitor`, which has three incoming transitions: transition $T_2(G_2, A_2)$, transition $T_3(G_3, A_3)$ from state `Battery_Control`, and transition $T_4(G_4, A_4)$ from state `Power_Low`, where $G_3 = [\texttt{true}]$ and $G_4 = [\texttt{true}]$. The guard of transition T_2 has been updated, hence we just change guards of transition T_3 and T_4 as $G_3 = [\texttt{upC}]$ and $G_4 = [\texttt{upC}]$ by applying **Integration Rule 3.2**.

V. Medical Ventilator Case Study

In this section, we perform a case study on the recalled medical ventilator scenario given in Section I. We demonstrate the differences of two models shown in Fig. 3 and Fig. 4.

A. Validation of System Models

We define a safety criteria as: the ventilator must raise a low power alarm before shutdown. In the system statechart models shown in Fig. 3 and Fig. 4, the safety criteria is expressed as: the state `B_Low_Alarm` must be activated before state `ShutDown`'s activation.

We take the scenario **S2** and **S4** in Example 1, i.e., $-10 \leq T < 15 \wedge 0.4 \leq H \leq 0.6$, as an example to show the validation of medical ventilator models with/without physical environment assumptions. The simulation results show that: (1) the safety criteria fails in the model without environment assumptions as shown in Fig. 5; and (2) the criteria is satisfied in the model with assumptions as shown in Fig. 6.

B. Formal Verification of System Models

For safety critical medical cyber-physical systems, validation is not adequate for guaranteeing their correctness and safety, and formal verification is required. In this paper, we use the Y2U[1] tool [23] to transform system models represented by Yakindu statecharts to UPPAAL timed automata for formal verification.

[1]The Y2U tool is available at www.cs.iit.edu/~code/software/Y2U.

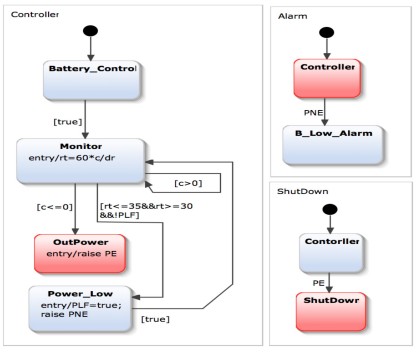


Fig. 5. Simulation Result of Ventilator Model without Assumptions

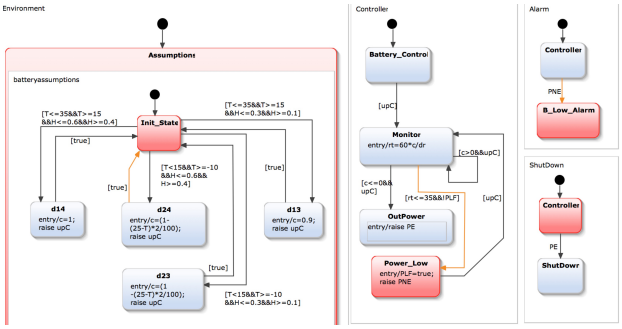


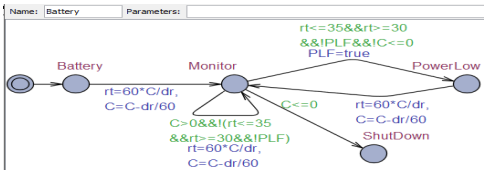
Fig. 6. Simulation Result of Ventilator Model with Assumptions

The UPPAAL models transformed from Yakindu statechart models by the Y2U tool are shown in Fig. 7. The safety criteria can be checked by the following formula in UPPAAL: $A[\ ]\ Battery.ShutDown\ imply\ PLF == true$. The verification results also show that: (1) the criteria fails in the model without considering physical environment assumptions (Fig. 7(a)); and (2) the criteria is satisfied in the model with assumptions (Fig. 7(b)).

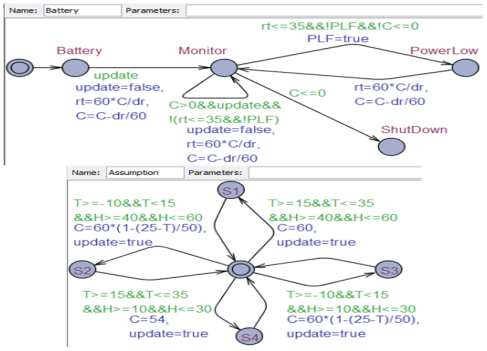
VI. Conclusion and Future Work

The paper presents a mathematical assumption model and its composition rules to explicitly specify physical environment assumptions of M-CPS. In addition, we introduce an approach to automatically transform assumption models to statecharts and provide strategies to integrate assumption statecharts with system models for validation and verification. The case study of a simplified medical ventilator scenario clearly demonstrates that modeling and integrating physical environment assumptions in M-CPS design can improve M-CPS safety.

In our case study, the specific safety property is satisfied in the integrated model. However it is possible that certain properties fail to hold during model verification. In this case, being able to trace back to the assumption model that causes the failed properties is important and is our immediate next step in the future work. Furthermore, M-CPS often not only involves physical environment assumptions, but also human behavior assumptions which is another yet to be explicitly modeled.



(a) Model without Assumptions



(b) Model with Assumptions

Fig. 7. UPPAAL Model

Acknowledgment

The research is supported in part by NSF CNS 1545008 and NSF CNS 1545002.

References

[1] Lui Sha and Jose Meseguer. Analytical system composition. In *The First Analytic Virtual Integration of Cyber-Physical Systems Workshop*, 2010.

[2] U.S. Food and Drug Administration. Medical device databases. http://www.fda.gov/medicaldevices/deviceregulationandguidance/databases/.

[3] U.S. Food and Drug Administration. Medical ventilators - faulty batteries. http://www.fda.gov/MedicalDevices/Safety/ucm460951.htm.

[4] Charles W. Kerechanin II, Protagoras N. Cutchis, Jennifer A. Vincent, Dexter G. Smith, and Douglas S. Wenstrand. Development of field portable ventilator systems for domestic and military emergency medical response, 2004.

[5] U.S. Food and Drug Administration. Electrostatic discharge may cause pump failure. http://www.fda.gov/MedicalDevices/Safety/ListofRecalls/ucm435811.htm.

[6] U.S. Food and Drug Administration. Applying human factors and usability engineering to medical devices. http://www.fda.gov/downloads/MedicalDevices/.../UCM259760.pdf.

[7] J. Goldmann. Medical device interoperability to enable system solutions at the sharp edge of healthcare delivery. In *White House Homeland Security Council Biodefense Directorate Conference*, Apr 2010.

[8] L. Schmitt, T. Falck, F. Wartena, and D. Simons. Novel iso/ieee 11073 standards for personal telehealth systems interoperability. In *In Proc. of Joint Workshop on HCMDSS-MDPnP*, 2007.

[9] Medical Device "Plug and Play" Interoperability Program. Ice standard: Integrated clinical environment. http://www.mdpnp.org/mdice.html.

[10] Medical Device "Plug and Play" Interoperability Program. Device clock synchronization. http://www.mdpnp.org/devicesynchronization.html.

[11] Cheolgi Kim, Mu Sun, Sibin Mohan, Heechul Yun, Lui Sha, and Tarek F. Abdelzaher. A framework for the safe interoperability of medical devices in the presence of network failures. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 149–158, 2010.

[12] Tirumala A.S. An assumptions management framework for systems software. In *Doctoral Thesis. University of Illinois at Urbana-Champaign*, 2006.

[13] S. Leue. Baby death due to software-controlled air bag deactivation? In *ACM Risks Digest*, March 1999.

[14] N. Leveson and C. Turner. An investigation of the therac-25 accidents. *IEEE Computer*, 26:1841, July 1993.

[15] Patricia Lago and Hans van Vliet. Explicit assumptions enrich architectural models. In *In Proceedings of the 27th international Conference on Software Engineering*, 2005.

[16] Lewis, T. A. G., Mahatham, and Wrage L. Assumptions management in software development. In *Technical Report. CMU*, 2004.

[17] R. Alur, D. Arney, E. L. Gunter, I. Lee, J. Lee, W. Nam, F. Pearce, S. V. Albert, and J. Zhou. Formal specifications and analysis of the computer-assisted resuscitation algorithm (cara) infusion pump control system. In *International Journal in Software Tools for Technology Transfer*, Apr 2004.

[18] D. Arney, R. Jetley, P. Jones, I. Lee, , and O. Sokolsky. Formal methods based development of a pca infusion pump reference model: Generic infusion pump (gip) project. In *In Proc. of Joint Workshop on HCMDSS-MDPnP*, Jun 2007.

[19] A. Ray and R. Cleaveland. Unit verification: the cara experience. In *International Journal on Software Tools for Technology Transfer*, 2004.

[20] R. Jetley, S. P. Iyer, and P. L. Jones. A formal methods approach to medical device review. *IEEE Computer*, 39(4), Apr 2006.

[21] Cadex Electronics. Battery university. http://batteryuniversity.com/.

[22] European Hearing Instrument Manufacturers Association. Ehima recommendations for zinc-air hearing aid batteries. http://www.ehima.com/wp-content/uploads/2014/03/EHIMA-Battery-Recommendations_V2.0.pdf.

[23] Chunhui Guo, Shangping Ren, Yu Jiang, Po-Liang Wu, Lui Sha, and Richard Berlin. Transforming medical best practice guidelines to executable and verifiable statechart models. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10, April 2016.