NATIONAL UNIVERSITY OF SINGAPORE

CS4211 Formal Methods for Software Engineering

Smart Fridge

Contents

Introduction

Both the industry and research sectors have been focusing on the development of smart fridges. Smart fridge integrates the usual traditional fridge with computer capabilities and having a WIFI connection to the Internet. A motivation for having smart fridges is to support the idea of ubiquitous computing, also known as Internet of Things. Currently, there are smart fridges / minibars being integrated with the hotel's Property Management System (PMS) to automatically track the items which hotel guests have taken out of the minibar thus reducing the usual manual work required.

With the complexities involved in developing a smart fridge, our project aims to model the smart fridge system, followed by verifying the system through a series of assertions.

This report will present the problems of existing fridges in the hospitality industry and how does our solution solve the problem raised by the industry. A detailed breakdown of the system modelling will be given in the later part of the report. Both the properties and processes of each features will be elaborated. Thereafter, we will analyze the result of our experimentation with the assertion of the system. Lastly, we will discuss and give feedback on the PAT system from our experience of using it.

Problem Description & Solution

Automated Purchase system

Hospitality-industry experts point out that revenue generated from selling the occasional $5 can of Heineken often don't make up for the cost of tending minibar in hundreds of hotel rooms. "Minibars are very labor-intensive: Hotels have to buy the items, stock them and then track the inventory," says Dale Turner, president of Turn Key Hotel Advisors, a hospitality consulting firm based in Dallas.

In Room Solutions, a minibar manufacturer in Wheeling, Ill., conducted a survey of clients to determine average daily revenue generated by one minibar and found profits ranged from $5.78 at a five-star hotel in New York City to $1.45 for a suburban three-star property.

The hotel's "Intelligent Rooms" are equipped with the latest in minibar technology: an "e-fridge" equipped with sensors that alert the central computer system at the hotel when an item is removed.

There is also less time to peruse the snack aisle: If an item is off its sensor pad for more than 30 seconds, it is automatically billed to the room. The Intercontinental says signs in the rooms alert guests to the rules.

Others concur. "The new technology makes the prospect of having a minibar much more profitable," says Philippe Duverger, vice president of marketing for Bartech Systems International, a Millersville, Md., company that supplies high-tech minibars to several major hotels, including the Intercontinental in Houston.

In the meantime, guests who have grown accustomed to using the minibar as their own personal refrigerator will find themselves at a loss for shelf space. The units are 25% smaller than their low-tech predecessors, and guests who inadvertently move items around will need to put them back on the correct sensor pads within the allotted time frame.

Expiry item check

Dale Turner says that in recent years, many of his clients purchasing hotels have chosen to remove minibars rather than deal with the hassle. "By the time you stock a refrigerator full of 40 different items, most of which have a limited shelf life, it adds up to a lot of time and money," Mr. Turner says. "People don't want their $8 Butterfinger to be stale."

If we were to check out the current App store for major platforms, there are several applications which aim to help consumer to manage their fridge, including keep track of the expiry dates of the items inside the fridge.

In addition, there is already now smart fridge offered by Samsung T9000 "smart" refrigerator with such a feature. Leftovers will have clear expiration dates on the semi-transparent surface of the smart fridge. It will calculate the freshness of food and automatically create a shopping list when we're missing or are low on essential items like milk.

Although the mobile app assist in the realm of personal home and the invention of smart fridge by Samsung for the home-consumer, there is no visible solution for the Business to Business industry, particularly the hospitality industry. That is where our smart fridge comes in.

When the housekeeping attendant places a perishable item inside our smart fridge, the fridge will record the expiry date of the item via our sensors. Everyday, the fridge will check all the expiry date of each of the item inside the fridge to ensure the freshness of the our perishable products. Once the fridge detect a expired items, it will alert the housekeeping department which item at what position has expired, and need to be replaced.

Through the sensors, the housekeeping attendants will also know which items have been taken out of the fridge. In this way, the attendants will be able to prepare in advance during their housekeeping round. This saves labour effort and prevents logistics nightmare for the hotel, hence increasing productivity during housekeeping.

Fridge security

We heard of the often complaints from hotel occupants that they had been charged for items which they did not take during their period of stay in the hotel.

With this problem in mind, our smart fridge sets to solve the theft problem. We have linked every single one of our smart fridge with one hotel room uniquely. There is a physical spring latch lock holding the fridge door and the main fridge compartment.

The fridge will remain locked physically in the hotel room unless our hotel guest insert his/her hotel room key card into the key card holder inside the hotel room. When the key card is inserted, the latch will be pulled back and unlocked the fridge. When the hotel guest leaves the room, the fridge will be locked again once he/she took out the key card from the hotel room key card holder. During housekeeping, the housekeeping attendants will be able to unlock the fridge with the universal hotel room key card.

With this additional layer of security added to the fridge, there is no way for external theft incident to happen. Then the hotel guests will not be able to complain of missing items in the fridge, nor they are able to complain of being charged to their room folio.

System Modeling

This section explains in detail the various components of the PAT program our team has delivered to test the feasibility of our smart fridge system.

The following assumptions of the system are:

- The fridge will be **fully stocked** before each occupant's stay.
- The Smart Fridge requires **power supply**
- Each holder is equipped with a **sensor** to detect item presence, as well as an LED to indicate status of restock or transaction.
- The inventory of the Smart Fridge is tracked by the hotel's **Property Management System (PMS)**.
- Fridge has a **maximum capacity** for the items.
- Without a the hotel room **key card** being inserted, the fridge door remains locked.

Table 3.1 illustrates the meaning of each variable that is being used in the PAT program and which component of the smart fridge system it models. Table 3.2 describes each constant that was used in the PAT program. Table 3.3 shows the events that models the smart fridge system. Table 3.4.

These variables, constants and events are further categorized in detailing how they work together in modelling each of the features - Automated Purchase System, Expiry Item Check and Fridge Security - which were illustrated in the "Problem Description & Solution" section. This is being shown in Table 3.5, Table 3.6 and Table 3.7 respectively.

Table 3.1- Variables

Component	Variable	Explanation
Fridge Power Supply	powerSupply	Indicates whether there is power supply in the room or not
Fridge Door Status	door	Indicates whether the door is opened or closed
Fridge Door Lock	lock	Indicates whether the fridge door is locked or unlocked
Room	room	Indicates whether the room is available or occupied
Key Card	keyCard	Indicates whether the hotel room key card is inserted or not inserted
User	user	Indicate whether the user is in the hotel room or not
Items' Position	itemPresence[i]	Represents the status of each item sensor in the smart fridge, whether an item is detected by the sensor or not
Items' Position	physicalItemPresence [i]	Represents the status of each item sensor in the smart fridge, whether an item is physically on the sensor or not
Sale of Item	itemSold[i]	Represents whether item that was on the sensor has been sold or not
Grace Period of Items	gracePeriods[i]	Indicates whether an item at item sensor i has been taken out for more than the grace period or not
Total items sold during the occupant's stay	totalSale	The number of items that have been sold
Remaining days before expiry	daysToExpire[i]	Indicates the number of days left before the current item expires

Expired item sold	expiredItemSold	Indicates whether an expired item has been sold
Discrepancy occurred	discrepancyOccurred	Indicates whether there has been a discrepancy in the number of items in the smart fridge

Table 3.2 - Constants

Constant	Definition
ON	fridge power is turned on
OFF	fridge power is turned off
OPEN	fridge door is opened
CLOSED	fridge door is closed
UNLOCKED	fridge door is unlocked
LOCKED	fridge door is locked
ROOM_AVAILABLE	hotel room is available
ROOM_BOOKED	hotel room is occupied
IN_CARD_HOLDER	room key card placed in the holder
NOT_IN_CARD_HOLDER	room key card not placed in the holder
IN_ROOM	user is in the room
NOT_IN_ROOM	user is not in the room
NUM_OF_ITEMS	number of items in the fridge
IN	item is in the fridge
OUT	item has been taken out of the fridge
SOLD	item in fridge has been sold
IN_STOCK	item in fridge has not been sold

WITHIN_GRACE_PERIOD	item’s grace period for being taken out of the fridge is still ongoing
OVER_GRACE_PERIOD	item’s grace period for being taken out of the fridge is up
DAYS_TO_EXPIRE	days for an item to expire upon placing the item into the fridge
INVALID	an item sensor position is not valid in a particular situation

Table 3.3 - Events

Event	Explanation
checkInHotelRoom	User checks into room (i.e. 1st day at hotel)
checkOutHotelRoom	User checks out of room (i.e. not staying in hotel anymore)
userEnterRoom	User enters the room
userExitRoom	User exits the room
insertKeyCardIntoHolder	Key card is being inserted into the card holder of the hotel room. If power supply is on, then the fridge door will be unlocked.
removeKeyCardFromHolder	Key card is being removed from the card holder of the hotel room. If power supply is on, then the fridge door will be locked.
openFridgeDoor	Open the fridge door if fridge door is closed and unlocked
closeFridgeDoor	Close the fridge door if fridge door is open and user is in the room
takeOutItemFromFridge	Item is being taken out of the fridge
putBackItemIntoFridge	Item is being placed back into the fridge when grace period is still ongoing
itemSoldOccurred	When item at sensor position i has been taken out of the fridge and grace period is up, item is considered sold
oneDayPass	Reduces the expiry days left for each item in the fridge by 1 and replenishes items that are expired or sold
oneWeekPass	Reduces the expiry days left for each item in the fridge by 7 and replenishes items that are expired or sold
gracePeriodTimerUp	If an item at sensor position i has been taken out of the fridge and the power supply is on, then grace period for that item is up

blackOut	The power supply to the fridge goes off
resume	The power supply to the fridge resumes and it checks for discrepancies for items currently in the fridge as compared to before the blackout
hungryForItem	A prefix event merely for the purpose of avoiding the unexpected restriction caused by atomic process

Table 3.4 - Macros

Macro	Explanation
sellItem (i)	When item at sensor position i has been taken out of the fridge and grace period is up, item is considered sold
replenish	Replenish items that are not in the fridge or are expired. During a blackout, when items are replenished, a discrepancy in the items are assumed to be check for manually and counted as an item sale.

Table 3.5 - Automated Purchase Feature

Events / Macros	Variable	Constant
checkInHotelRoom checkOutHotelRoom	room	ROOM_AVAILABLE ROOM_BOOKED
	keyCard	IN_CARD_HOLDER NOT_IN_CARD_HOLDER
	user	IN_ROOM / NOT_IN_ROOM
	door	CLOSED
	totalSale	0
userEnterRoom userExitRoom	room	ROOM_AVAILABLE / ROOM_BOOKED
	user	IN_ROOM / NOT_IN_ROOM
openFridgeDoor closeFridgeDoor	door	OPEN / CLOSED
	lock	UNLOCKED / LOCKED
	user	IN_ROOM / NOT_IN_ROOM
takeOutItemFromFridge putBackItemIntoFridge	itemPresence[i] physicalItemPresence[i]	IN / OUT
	door	OPEN /CLOSED
	powerSupply	ON / OFF
	user	IN_ROOM / NOT_IN_ROOM
	gracePeriod[i]	WITHIN_GRACE_PERIOD/

		OVER_GRACE_PERIOD
gracePeriodTimerUp	gracePeriod[i]	WITHIN_GRACE_PERIOD/ OVER_GRACE_PERIOD
	powerSupply	ON / OFF
	daysToExpire	{-1,0,1...N} where N = DAYS_TO_EXPIRE -1 = INVALID
itemSoldOccurred / sellItem (i)	totalSale	{0,1..N} where N = NUM_OF_ITEMS
	itemSold[i]	SOLD / IN_STOCK
	daysToExpire[i]	{-1,0,1...N} where N = DAYS_TO_EXPIRE -1 = INVALID
	expiredItemSold	any number >= 0

Table 3.6 - Expiry Checking Feature

Events / Macros	Variable	Constant
replenish	daysToExpire[i]	{-1,0,1...N} where N = DAYS_TO_EXPIRE -1 = INVALID
	itemPresence[i] physicalItemPresence[i]	IN / OUT
	itemSold[i]	SOLD / IN_STOCK
oneDayPass	daysToExpire[i]	{-1,0,1...N} where N = DAYS_TO_EXPIRE -1 = INVALID
	itemPresence[i]	IN / OUT
	powerSupply	ON / OFF
oneWeekPass	daysToExpire[i]	{-1,0,1...N} where N = DAYS_TO_EXPIRE -1 = INVALID
	itemPresence[i]	IN / OUT
	powerSupply	ON / OFF

Table 3.7 - Security Feature

Events	Variable	Constant
blackOut resume	powerSupply	ON / OFF
	itemPresence[i] physicalItemPresence [i]	IN / OUT
	discrepancyOccurred	true / false

insertKeyCardIntoHolder removeKeyCardFromHolder	keyCard	IN_CARD_HOLDER / NOT_IN_CARD_HOLDER
	powerSupply	ON / OFF
	lock	UNLOCKED / LOCKED

To prevent any uncertainties regarding the system modelling, the following are some points to note regarding the scope of modelling.

- Expired items and unoccupied item sensors are being replenished with new items when a day has passed.
- During replenishment, we assume it is a manual human process that resets the fridge's stock to normal. Hence, any stock discrepancies that are detected by humans is not considered as a discrepancy detected by the fridge, as it is outside of its capability scope.
- All discrepancies detected will be considered as sold.
- When power has been resumed from a blackout and after discrepancies in items' position have been checked for, the missing items will be replenished in the fridge.
- When user checks out, the key card must be with the user and not in the room's key card holder.

Investigated Properties

Given the model that we have previously elaborated on our system, we will like to examine on the correctness and completeness of our model. Below is some of the properties that we have come up with to test the system for its vulnerabilities:

Resume after blackout

Whenever a blackout has occurred in the hotel room, the power supply to the hotel room will eventually resume, regardless of other conditions such as the fridge, the occupant and the hotel room.

Assertion in P.A.T: #assert Asystem |= [](blackOut -> <> resume);

Checkout after check in

Whenever a person checks into the hotel room, he/she will eventually check out that same hotel room, regardless of other conditions such as the fridge and the hotel room.

Assertion in P.A.T: #assert Asystem |= (checkInHotelRoom -> <> checkOutHotelRoom);

Grace period time up after taking out an item

If an occupant is interested to purchase an item from the fridge, he/she will take the item out from the fridge. The fridge will eventually notify this intention when the grace period for the item to be put back is over.

Assertion in P.A.T: #assert Asystem |= (takeOutItemFromFridge -> <> gracePeriodTimerUp);

No expiry item sold

An expired perishable item can never be sold from the fridge since any expired item in the fridge will be replenished with each passing day.

Assertion in P.A.T: #assert Asystem |= [] noExpiredItemCanBeSold;

No Item resale during stay

All the items cannot be sold again when the item has already been sold once. It includes the situation when the occupant purchased an item from the fridge via the convenient method. After that, he put the item back into the same position he took out, and took it out again.

Assertion in P.A.T: #assert Asystem |= [](!itemSaleDecreased);

No item sale after checkout

After an occupant has checked out from a hotel room, it is not possible for any item sale again.

Assertion in P.A.T: #assert Asystem |= [](checkOutHotelRoom -> !itemSoldOccurred);

Item sale always notified

The fridge will always notify the property management system whenever there is a sale, regardless if the sale is detected via the convenient method of taking out a item from the fridge, or any other method.

Assertion in P.A.T: #assert Asystem |= [](itemSoldOccurred -> <> itemSaleIncreased);

Close fridge door when locked

The fridge door still can be closed even if the fridge is locked.

Assertion in P.A.T: #assert Asystem |= (doorIsLockedButOpened -> X closeFridgeDoor);

Take out item without key card

An occupant is able to take out an item from the fridge when the key card is not inside the key card slot of the respective hotel room.

Assertion in P.A.T: #assert Asystem |= (fridgeDoorOpenedButKeyCardNotInSlot -> X takeOutItemIntoFridge);

Put back item without key card

Similar as above, an occupant is able to put an item back into the fridge when the key card is not inside the key card slot of the respective hotel room.

Assertion in P.A.T: #assert Asystem |= (fridgeDoorOpenedButKeyCardNotInSlot -> X putBackItemIntoFridge);

Item sale after open door

There will never be item sold (regardless of discrepancy) until the fridge door is opened

Assertion in P.A.T: #assert Asystem |= (openFridgeDoor R noItemSold);

Report discrepancy during blackout

Discrepancy will only occur under the condition where there was a blackout occurred

Assertion in P.A.T: #assert Asystem |= (blackOut R noDiscrepancy);

All item sold

A room occupant can buy all the items inside the fridge at a point of time during his stay in the hotel room.

Assertion in P.A.T: `#assert Asystem reaches allItemsSold;`

Exit room when door open

An occupant can leave the room during his period of stay without closing the fridge door.

Assertion in P.A.T: `#assert Asystem reaches exitRoomWithFridgeDoorOpen;`

Deadlock-free

The program is deadlock free.

Assertion in P.A.T: `#assert Asystem deadlockfree;`

As you can see from the assertion that was used, we have used a variety of assertion methods on the modelling. We have used the most basic reachability assertion to using the LTL assertions with keywords such as X (next) and R (release).

Design Decision

Handling the case of discrepancy check

We encounter an issue when we are modelling the security feature of the fridge in detection of the mismatch of the number of item before and after the blackout. We named such a mismatch an discrepancy. The detection of the discrepancy come in after the power resume for the hotel room after the blackout. It is naturally the only period of time to detect for discrepancy as it is the first instance when the electronic-operated smart fridge is able to check. There are two possible ways to modelling the discrepancy check:

Model the itemPresence as an "electric" sensor

Problem: As the fridge can only detect changes when the power supply is running, there can be some false negative case. For example, if a blackout occur and the fridge door is open, we are not able to model the case when the occupant takes out a item from the fridge as the electric sensor cannot function without electricity

Model the itemPresence as an "physical presence" sensor

As a physical presence sensor, the system only record the presence of the fridge into an electric memory state just before the blackout.

Problem: The approach is not as realistic since a blackout can occur unexpectedly at any point of time. Hence in reality the fridge will not be able to record the state of itemPresence just before blackout as it is not able to predict blackout.

Decision

After discussion, we model the discrepancy check with itemPresence as an "electric sensor". In addition, to resolve the problem stated above, we implement another array to denote the physical presence of the items. Hence in the solution submitted, whenever a customer take out an item from the fridge, both the array for the electric sensors and physcial presence of the itemPresence will be affected, hence it is inline with modelling result we want to express. Such an approach is at the same time realistic.

Modelling with atomic process

An atomic process is required for item_action(i) to ensure that there is no interruption between the time when an item is taken out of the fridge to when the grace period timer is up. Reason being that without it, it introduces implications. Such implication could be that when the item at position 1 is taken out, physicalItemSensor[1] = OUT and if a blackout occurs and resumes before the gracePeriodTimerUp.1 event gets to execute. It detects a discrepancy and will count the item as sold but finally when gracePeriodTimerUp.1 is executed, this event occurrence will be meaningless as the item has already been considered sold.

However, using an atomic process in PAT would cause the process to have higher priority thus allowing it to execute continuously in a loop. This has been rectified by having the event hungryForItem.i before the atomic process is executed so that other processes will have a fair chance of being executed.

Modelling of Assertions that contain conditions as pre-condition

There are some assertions which we made in this project that require certain condition to be fulfilled before we can check for its correctness in executing the event.

For example, in the assertion “Take out item without key card”, we will need to simulate the condition:

- The key card is not inside the key card holder
- The user is inside the room
- The fridge door is open

In modelling such an assertion, we have used the approach to first define a condition to fulfill all the pre-requirement of assertions. Only after the condition is defined, we check that the condition will lead to an desire event happening.

Real Time System Modelling

The fridge system has time factors involved in its regular operation. Additionally, our modelling of the hotel environment require us to use time factor to model the real life situation realistically. One of the time factors is the grace period to calculate the duration between the item being taken out from the fridge and the item being considered as sold. The other time factor is timed process which we used to simulate an artificial period of time passed by so that the features of expiry checking and replenish can be modelled.

Hence it is logically to model the system using the Real Time System Modelling module. However, as we investigate, we realise that it is unfeasible to use RTS modeling as Wait is atomic. In our system modelling, we allow other event to happen while the grace timer is happening. Wait violates the modelling that we wanted to build, hence we decided to adopt the RTS modelling module.

Macro modelling

Even though a macro is not considered as an event, it provides a neater and better understanding of the code. The events that we modelled as macros instead are sellItem(i) and replenish because they are often reused in different parts of the code. The events gracePeriodTimerUp, resume, oneDayPass and oneWeekPass usually lead to sellItem(i) being executed and checkInHotelRoom, oneDayPass and oneWeekPass will cause the execution of replenish.

System modelling

Our system processes are being executed by interleaving (|||) instead of parallel composition (||) because our processes do not have common events that can be synchronized. Furthermore, parallel composition does not include events which does data operations on global variables since this may lead to race conditions.

Experimentation

Due to the design restriction of our model, we have come up with a few versions of our original model. These versions further push the limitation of our model and allow the model to have a more realistic representation to the system modelling.

The number of items inside the fridge is kept constant at 2 throughout the investigation of the series of experimentation we carried out.

The final PAT model that we have submitted is the control condition since each experiment needs a control condition to compare against.

Experiment 1

Experiment 1 forces every data operation as an unnamed event and be prefixed by an event. In this way, the assertion can be more accurately modelled.

Observation

We observed that there is no difference in the number of transitions and visited states in the first 3 assertions. However, as we examine the other assertions, we noticed that verification statistics increased dramatically compared to the control. It seemed that the number of states visited, memory usage and total transition increased exponentially with the increased difficulty of the assertions.

In certain assertions, state explosion occurs and we are not able to verify the assertions. In such a case, we are unable to determine the result of the assertions against our control. Further assertion is needed to determine the result, such as more computing power of the machine running the PAT application.

Experiment 2

This experiment examines the effect of the power_supply_status() process and time_passed() process to the state space of the system. We restrict it such that blackout and time passed can only happen when someone has checked in the hotel room. As we can see from the result, the state space and the total transition does not increase by much.

Observation

We have observed that the verification statistics is lesser than what we have seen in the control. It is aligned with our expectation as we expected that more restrictive condition will result in less stated being visited.

Experiment 3

This experiment test with the restrictiveness of the condition and its respective assertions. In this case, we have removed the restriction that door must be open when we are checking for the assertions *"Take out item without key card"* and *"Put back item without key card"*.

Observation

There are some notifiable trends which we observe from the assertions. As aligned

"Take out item without key card" and *"Put back item without key card" :* So can concluded a less restrictive condition does not affect the performance of the assertions.

Possible Extension

After we have modelled the smart fridge system, we have observed a number of improvements that we can implement to make the smart fridge more user-friendly.

Automatic Door Close

As we observed in the assertions, the fridge door can remain open when the occupant leave the room. This may leave to wastage of electricity in such situations. In avoidance of such wastage, we will propose a new feature of the fridge to allow the door to close by itself after a certain time period. Such saving in electricity wastage will translate to less expenditure. This can benefit the hotel in earning more profit with decreased expenditure.

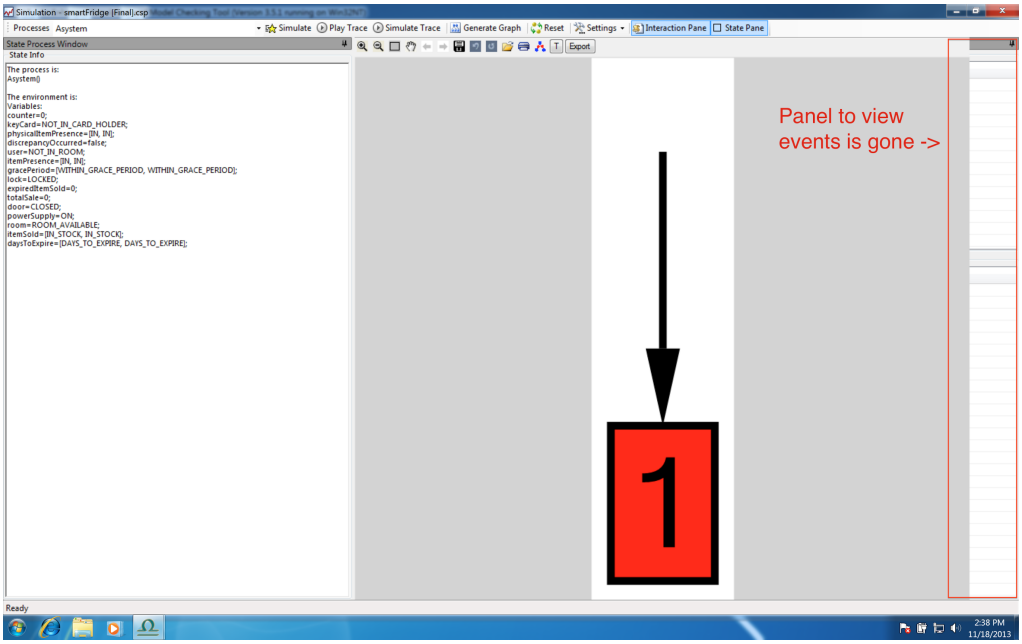
Administrator mode

As replenish occur daily during the hotel occupant stay, we will like to add a security feature to the fridge to further enhance the security of the fridge. The housekeeper has to enter the password to enter the administrator mode of the fridge. The system only allow replenish to happen in the administrator mode, hence external personnel will not be able to tamper with the fridge system.

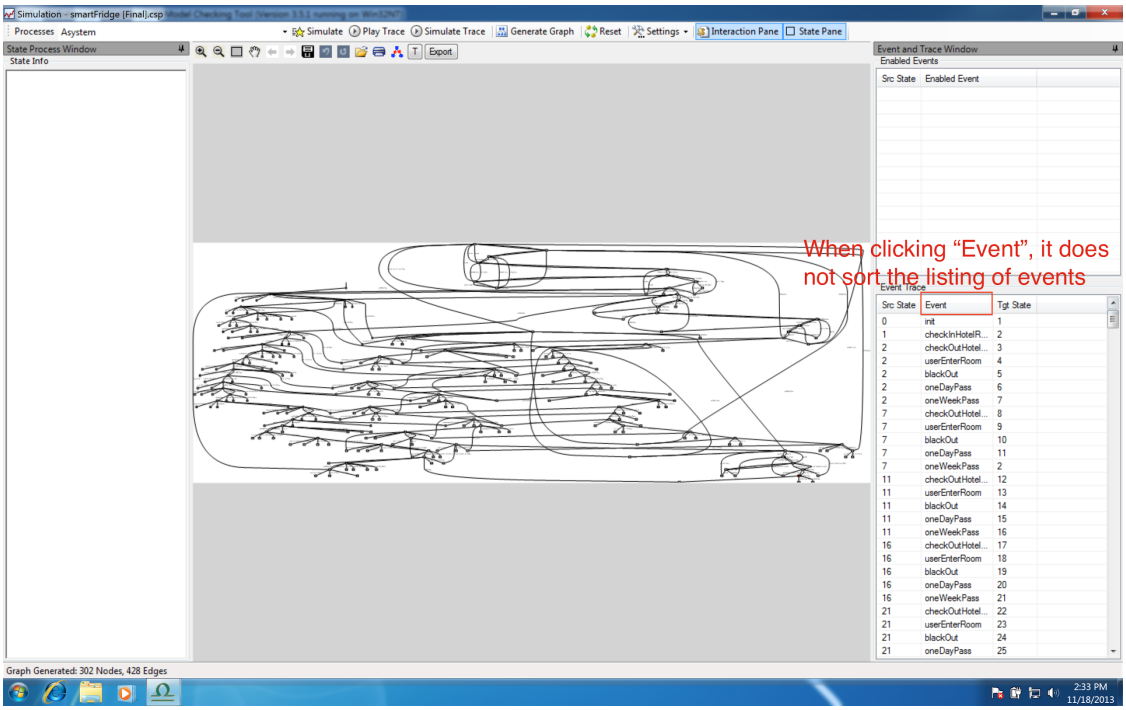
Feedback and Suggestion for PAT

Bug discovered

The right visual pane disappears when the window is minimized and then maximized again.



When “Event” on the right visual pane has been clicked, it does not sort the listing of events. This would be useful when checking our model’s behaviour base on an event (e.g. check if a certain event has occurred).



Limitation on PAT:

- Event synchronisation cannot occur between the same events that carry out data operations.
- No way to make an event happen immediately following another event, except the use of atomic block.

Conclusion

In summary, we have asserted that the fridge model is feasible in a way that it is able to automatically and correctly record the number of items sold and it will not allow an item to be unsold even if the user has placed back the item into the fridge when the grace timer is up. Furthermore, upon resuming power from an unexpected blackout, the fridge is able to discover discrepancies in the items from the time of blackout to the time of power resume. Such properties of the fridge is crucial to the hotel's business so as to ensure that there are no loopholes for the customer to consume the items from the fridge for free.