AGH

Systemy Czasu Rzeczywistego

Dokumentacja Projektowa

Winda Towarowa

Autorzy:
Michał Janiec
Bartosz Polnik

Prowadzący:
Dr. Inż. Michał Turek

29 listopada 2013

1 Wprowadzenie

Projekt został zrealizowany w ramach zajęć z przedmiotu Systemy Czasu Rzeczywistego. Jego celem jest zaznajomienie się z zagadnieniem tworzenia systemów czasu rzeczywistego w środowisku IBM Rational Rhapsody. Realizowanym przez nas przykładem jest winda towarowa.

2 Przedstawienie problemu

Winda towarowa to urządzenie stosowane w przemyśle. Jest szeroko stosowana w kopalniach, halach produkcyjnych czy restauracjach. Zastosowanie jest proste: Na jednym z pięter pracownik przywołuje windę, ładuje towar, a następnie wysyła windę na inne piętro, gdzie ktoś inny odbiera towar. W przeciwieństwie do windy osobowo towarowej nie przewozi się wewnątrz osób, co prowadzi do kilku uproszczeń; Sterowanie windą odbywa się w całości przy pomocy paneli sterowniczych znajdujących się na zewnątrz windy. Ponadto nie ma potrzeby stosowania automatycznie otwieranych drzwiczek. Konieczne jest także zachowanie wszelkich norm bezpieczeństwa: winda dba o zachowanie odpowiedniej prędkości podczas wznoszenia i opuszczania, jak i odpowiedniej wagi załadunku. Ponadto jak większość urządzeń przemysłowych windy towarowe często posiadają tak zwany kill switch - przycisk pozwalający na natychmiastowe wyłączenie urządzenia.

Pewna winda towarowa. Na panelu widoczny czerwony przycisk "kill switch"

3 Wymagania

3.1 Wymagania funkcjonalne

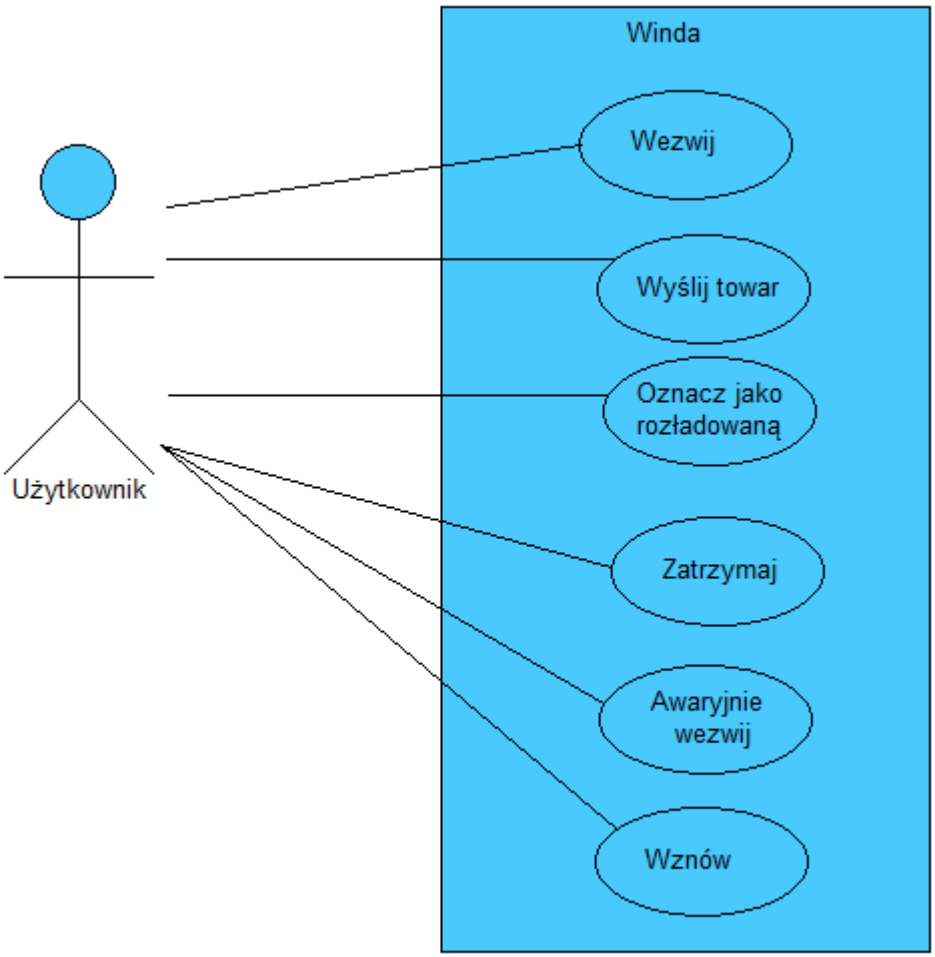
- przemieszczanie towarów między piętrami

3.2 Wymagania niefunkcjonalne

- niezawodność - brak przerw w działaniu a także poprawne zachowanie w sytuacjach brzegowych (np. gdy po wznowieniu zasilania winda znajduje się między piętrami)
- bezpieczeństwo - wykrywanie przeciążeń, niezamkniętych drzwiczek, możliwość natychmiastowego wyłączenia urządzenia

- prosty interfejs użytkownika - pozwoli uniknąć pomyłek - zwiększy bezpieczeństwo

4 Przypadki użycia



Scenariusze

- Wezwij windę
  - warunek wstępny - winda nie jest w stanie emergency
  - warunek końcowy - winda znajduje się na piętrze wzywającego
  - przebieg główny:
    * użytkownik wciska przycisk **bring here**,

        * użytkownik czeka.

- Wyślij windę
  - warunek wstępny - winda jest na tym samym piętrze co użytkownik, nie jest przeciążona, a drzwiczki są zamknięte
  - warunek końcowy - winda znajduje się na piętrze, na które została wysłana
  - przebieg główny:
    * użytkownik wciska przycisk z numerem piętra
    * użytkownik czeka.
- Oznacz windę jako rozładowaną - scenariusz podstawowy
  - warunek wstępny - winda jest na tym samym piętrze co użytkownik, nie jest przeciążona, a drzwiczki są zamknięte
  - warunek końcowy - winda może zostać wezwana przez kogoś innego
  - przebieg główny:
    * użytkownik wciska przycisk **ready**,
- Zatrzymaj
  - warunek wstępny - brak,
  - warunek końcowy - winda się zatrzymuje
  - przebieg główny:
    * użytkownik wciska przycisk **stop**,
- Wezwij awaryjnie
  - warunek wstępny - brak.
  - warunek końcowy: Winda znajduje się na parterze
  - przebieg główny:
    * użytkownik wciska przycisk **Emergency Bring Here**,
- Wznów
  - warunek wstępny - winda jest w stanie emergency,
  - warunek końcowy - winda oczekuje na wezwania
  - przebieg główny:
    * użytkownik wciska przycisk **Emergency ready**,

5 Architektura

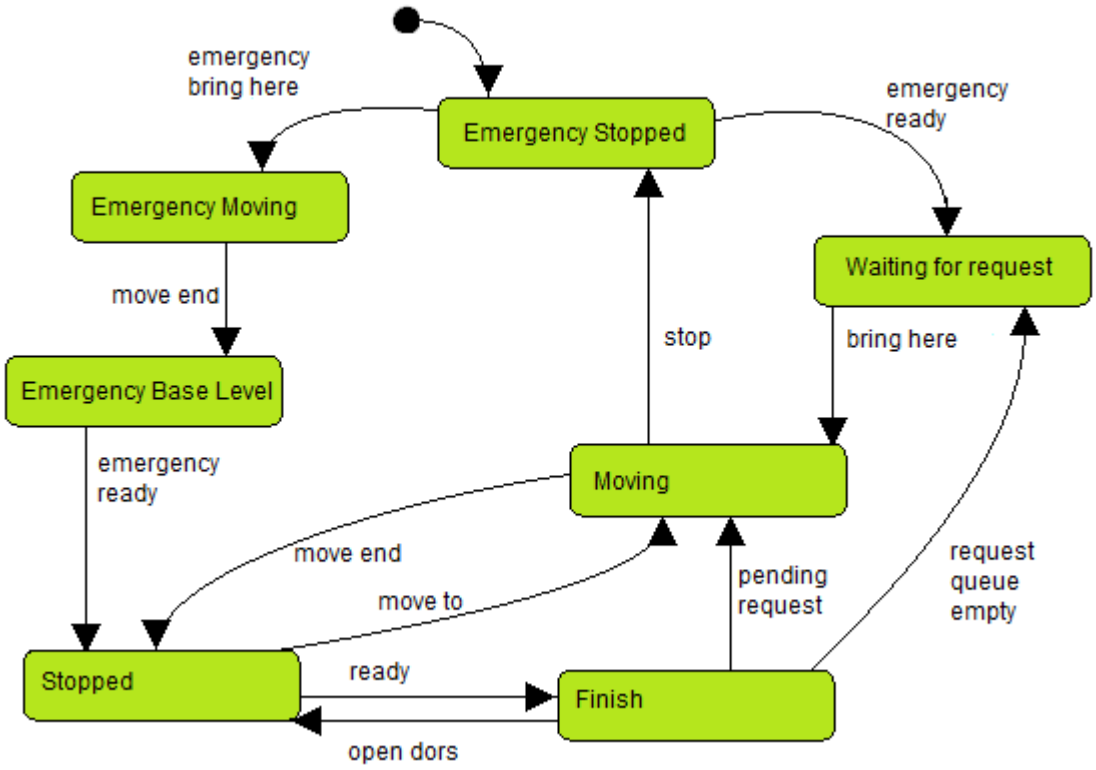
Winda została zaprojektowana jako maszyna stanowa.
Winda posiada detektory:

- detektor stanu drzwiczek (otwarte/zamknięte)
- detektor obciążenia windy

a także efektor:

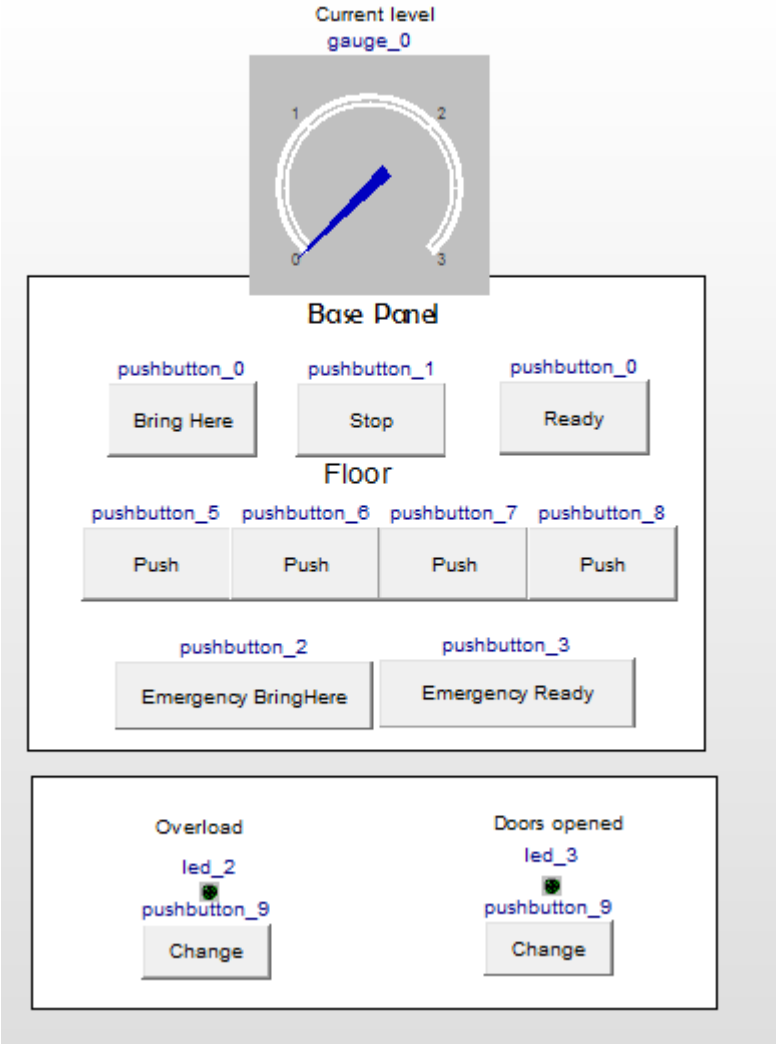
- silnik windy - przemieszczanie góra / dół

Uproszczony diagram stanów



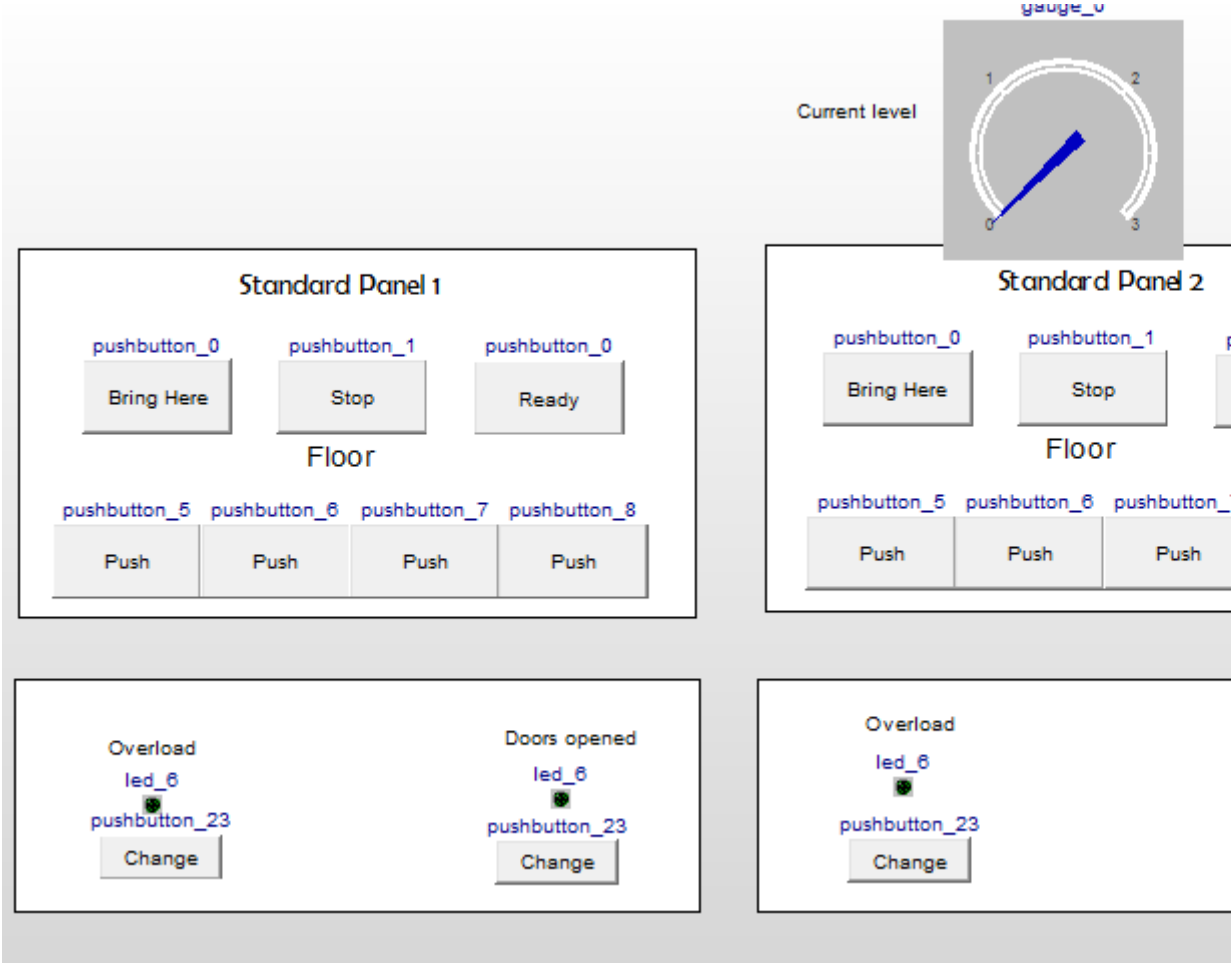
6 Interfejs graficzny

6.1 Interfejs wewnętrzy Rhapsody



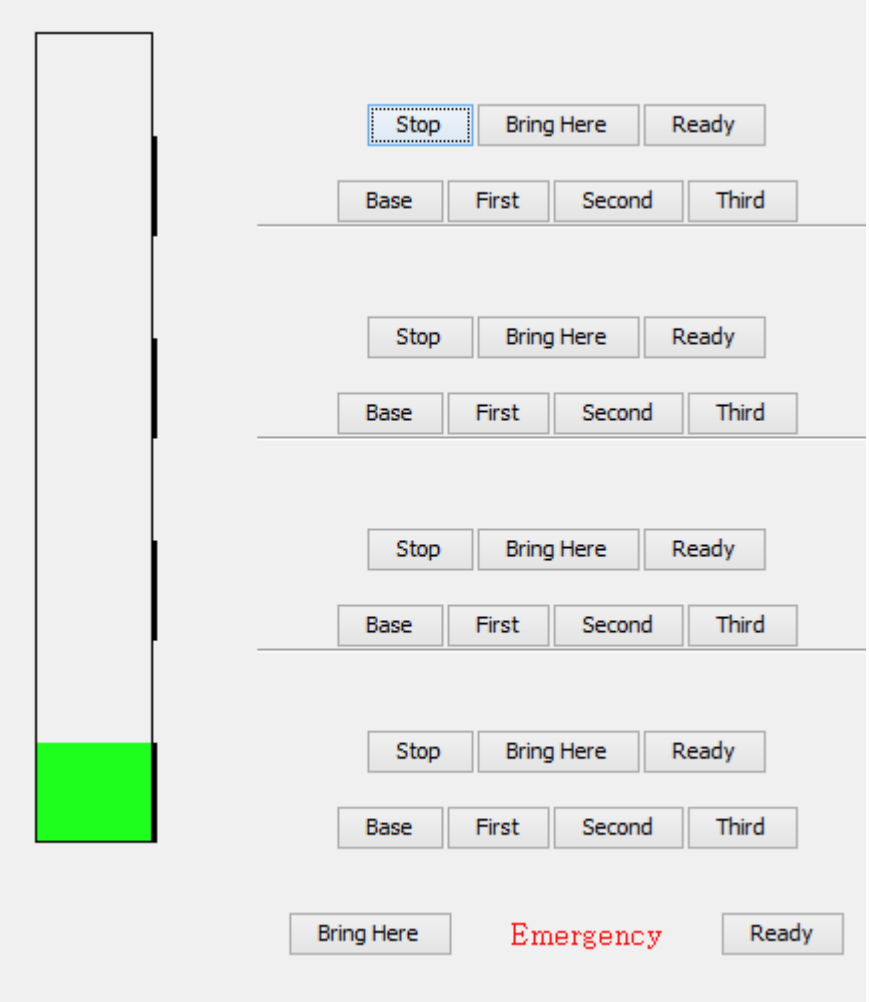
Interfejs na parterze

Wskazówka na górze pokazuje numer piętra. Na samym dole znajdują się diody informujące o tym czy drzwiczki są otwarte, oraz czy winda jest przeciążona. Jeśli chcemy zasymulować zmianę jednego z tych parametrów wystarczy kliknąć na przycisku change pod spodem. Środkowa część panelu to główny panel sterowniczy. Zawiera on przyciski **Bring here** do wzywania windy, **Stop** do zatrzymania, **Ready** do zakończenia akcji. Cztery przyciski poniżej służą do wysyłania na odpowiednie piętro. Natomiast przyciski emergecy służą do obsługi stanu emergency.



Interfejsy na pozostałych piętrach (interfejs 3 piętra nie widoczny) Panele na pozostałych piętrach są bardzo zbliżone. Nie posiadają jedynie przycisków **Emergency Bring Here, Emergency Ready**. Przykład użycia windy. Po uruchomieniu winda jest w stanie **Emergency Stopped**. Aby móc ją używać należy wcisnąć przycisk **Emergency Ready**. Winda jest gotowa do użytku. Teraz można kolejkować dla niej zadania. Aby rozpocząć zadanie należy na dowolnym piętrze wcisnąć **Bring here**. Winda podjedzie na odpowiednie piętro. Tam można ją załadować. Następnie posyłamy windę na odpowiednie piętro, gdzie może być rozładowana. Po rozładowaniu klikamy **Ready** i winda jest gotowa do przetworzenia następnego wezwania.

6.2 Interfejs zewnętrzny



Interfejsy zewnętrzny

Interfejs jest ódbiciem"tego z Rhapsody z tą różnicą, że bardziej czytelnie prezentuje stan windy. Różni się także sposób otwierania drzwiczek i przeciążania windy. W celu przeciążenia windy wystarczy kliknąć jej rysunek (kolor odwzorowuje stan). W celu otwarcia drzwi wystarczy kliknąć na ich rysunek.

7 Testy

7.1 Testy niezależności zmiennych które powinny być niezależne

Do testów wybraliśmy use case: **wezwij awaryjnie**. Zdecydowaliśmy się na niego, ponieważ winda posiada 3 podstawowe parametry: Stan drzwi

(otwarte czy zamknięte), stan obciążenia (czy przeładowana), położenie (wysokość). Znakomita większość operacji sprawdza wszystkie te stany więc pozostałe zmienne ograniczały by się do wartości interfejsu **Effectors** oraz ewentualnie kolejki zadań **Requests**. Oto lista zmiennych które nie powinny mieć wpływu na use case **wezwij awaryjnie**: wraz z klasami równoważności możliwych wartości dla tych zmiennych

Elevator.effectors	null	referencja	
Elevator.moving	true	false	
Elevator.requests	pusta	1 żądanie	5 żądań
LoadController.isOverloaded	true	false	
DoorsController.areDoosOpen	true	false	
Floor[0].areDoorsOpen	true	false	
Floor[1].areDoorsOpen	true	false	
Floor[2].areDoorsOpen	true	false	
Floor[3].areDoorsOpen	true	false	

Następnie utworzyliśmy 3 test case'y:

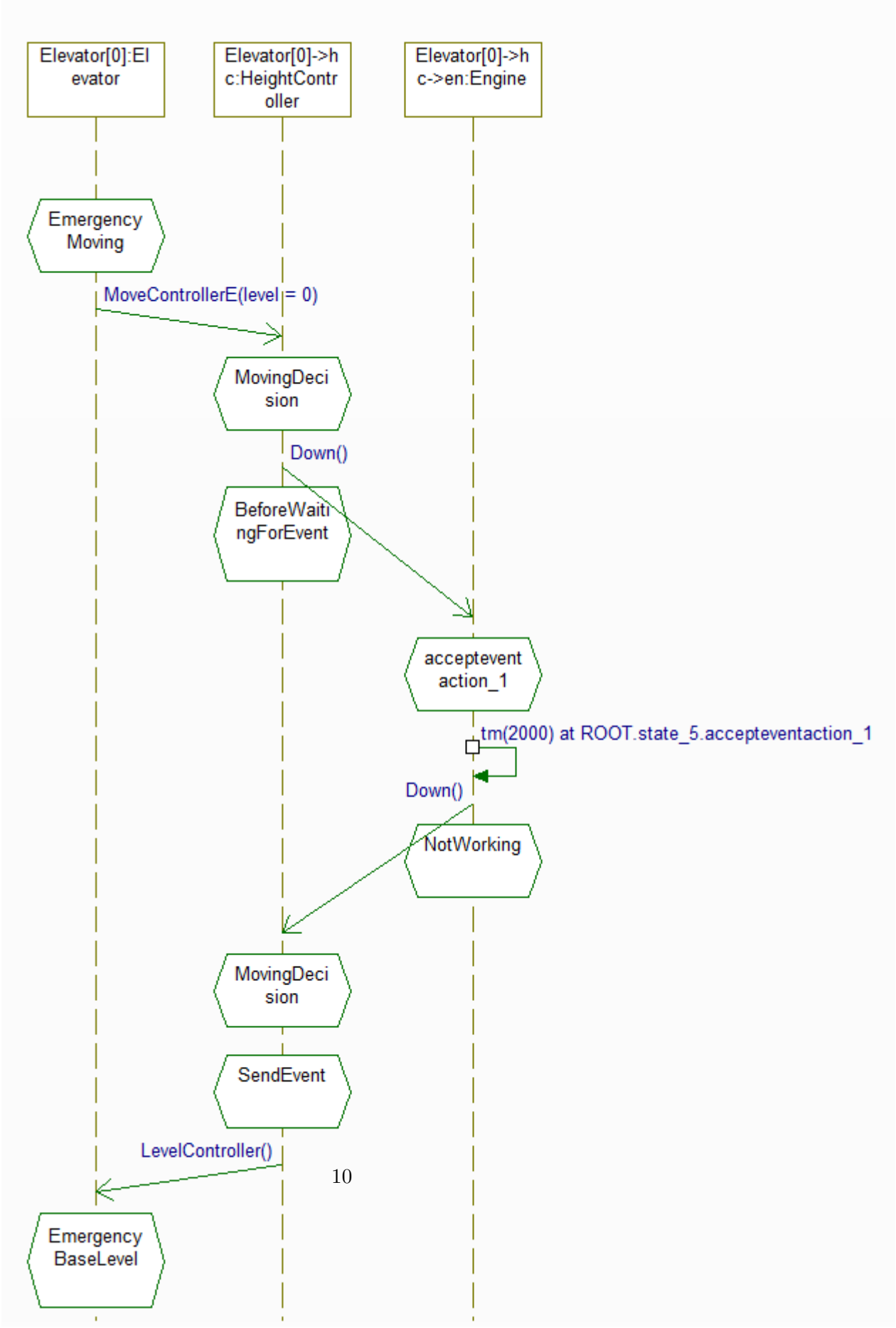
Zmienna	**wartość TC1**	**wartość TC2**	**wartość TC3**
Elevator.effectors	null	referencja	null
Elevator.moving	true	false	false
Elevator.requests	pusta	1 żądanie	5 żądań
LoadController.isOverloaded	true	false	true
DoorsController.areDoosOpen	true	false	true
Floor[0].areDoorsOpen	true	false	true
Floor[1].areDoorsOpen	true	false	true
Floor[2].areDoorsOpen	true	false	false
Floor[3].areDoorsOpen	true	false	true

Kolejną czynnością było przygotowanie diagramu sekwencji. Aby mieć taką możliwość należy najpierw doprowadzić windę do odpowiedniego stanu. W tym celu należy wykonać następujące kroki:

- Przywołać windę na dowolnym piętrze
- zatrzymać windę w czasie jazdy

Następnie aby wywołać omawiany use case należy wcisnąć przycisk **Emergency Birng Here**.

Przygotowaliśmy wzorcow sequence chart:


Elevator[0]:El evator
Elevator[0]->h c:HeightContr oller
Elevator[0]->h c->en:Engine
Emergency Moving
MoveControllerE(level = 0)
MovingDeci sion
Down()
BeforeWaiti ngForEvent
acceptevent action_1
tm(2000) at ROOT.state_5.accepteventaction_1
Down()
NotWorking
MovingDeci sion
SendEvent
LevelController()
Emergency BaseLevel

Następnie porównaliśmy sequece chart'y otrzymane w trzech powyższych przypadkach testowych każdo razowo otrzymując taki sam seqence chart. Można zatem pokusić się o wniosek iż zmienne które postulowaliśmy jako niezależne są takie w istocie.

7.2 Scenariusze

Podczas testowania windy należy zwrócić uwagę na wszelkie sytuacje wyjątkowe. Utworzyliśmy następujące scenariusze testowe:

7.3 Uruchomienie windy

1. Próba wezwania windy na pierwszym piętrze
2. Winda nie poinna się poruszyć
3. Wciśnięcie przycisku **Emergency Ready**
4. Próba wezwania windy na pierwszym piętrze
5. Winda powinna podjechać na pierwsze piętro

7.4 Załadunek windy

Warunek początkowy - winda jest poprawnie uruchomiona (przycisk Emergency Ready został już wciśnięty) - znajduje się na piętrze 0

1. Wciśnięcie bring here na parterze
2. Otwarcie drzwiczek,
3. Próba wysłania windy na następne piętro,
4. Winda nie powinna się poruszyć
5. Zwiększenie obciążenia powyżej progu,
6. Zamknięcie drzwiczek
7. Próba wysłania windy na następne piętro
8. Winda nie powinna się poruszyć
9. Zdjęcie ciężaru poniżej limitu
10. Próba wysłania windy na następne piętro
11. Winda powinna pojechać

7.5 Wzywanie windy i oznaczanie jako rozładowanej

Warunek początkowy - winda jest poprawnie uruchomiona, znajduje się na poziomie 0

1. wciśnięcie przycisku **bring here** na pierwszym piętrze
2. winda powinna przyjechać na pierwsze piętro
3. wciśnięcie przycisku **bring here** na poziomie 0
4. winda nie powinna podjechać (powinna czekać na rozładowanie)
5. wciśnięcie przycisku **Second** na pierwszym poziomie
6. winda powinna podjechać na drugie piętro
7. wciśnięcie przycisku **ready** na drugim piętrze
8. winda powinna rozpocząć kolejne zadanie (zjechać na poziom 0)

7.6 Przycisk stop i Emergency Ready

Warunek początkowy - winda jest poprawnie uruchomiona, znajduje się na poziomie 0

1. wciśnięcie przycisku stop
2. próba wysłania windy na pierwsze piętro
3. próba przywołania windy na pierwszym piętrze
4. winda nie powinna się poruszyć
5. przyciśnięcie przycisku **Emergency Ready**
6. próba przywołania windy na pierwszym piętrze
7. winda powinna się pojechać
8. wciśnięcie przycisku stop, gdy winda znajduje się między piętrami.
9. winda powinna się zatrzymać

7.7 Przycisk EmergencyBringHere

1. pozostawienie windy między piętrami (opisane w poprzednim teście)
2. wciśnięcie przycisku **Emergency Bring Here**
3. winda powinna zjechać na dolne piętro

8 Podsumowanie

Budowa oprogramowania przy pomocy środowiska IBM Rhapsody okazała się ciekawym i bardzo rozwijającym doświadczeniem. Produkt firmy IBM umożliwia modelowanie systemów przy pomocy różnorodnych diagramów, najważniejszym dla nas typem diagramu jest diagram stanów, który pozwala definiować i nazywać pewne momenty w czasie życia systemu oraz określać przejścia pomiędzy nimi. Dzięki możliwości wizualnej budowy i analizy systemów opartych o maszyny stanowe pakiet ten idealnie nadaje się do budowy systemów czasu rzeczywistego, zapewniając wysoką przejrzystość, pozwala unikać błędów i budować niezawodne oprogramowanie. Kolejną zaletą Rhapsody jest zastosowanie języków C++ oraz Java które są powszechnie znane i posiadają bardzo duże wsparcie techniczne i ogromne ilości bibliotek. Dzięki IBM Rhapsody udało nam się stosunkowo szybko stworzyć funkcjonalny i stabilny projekt.