**Automatyczna kasa biletowa – model architektury w języku AADL**

**2. Opis modelowanego systemu**

**2.1. Opis ogólny**

Model architektury AADL przedstawia uproszczoną automatykę stacjonarnej kasy biletowej, składającą się z czterech podstawowych komponentów:

* **User\_Interface (ui)** – interfejs użytkownika z panelem dotykowym do wyboru i potwierdzania parametrów biletu oraz do wyświetlania komunikatów.
* **Ticket\_Processor (tp)** – jednostka przetwarzająca, która odbiera żądania z interfejsu, zarządza logiką transakcji i steruje modułami płatności oraz drukarki.
* **Payment\_Unit (pay)** – urządzenie odpowiedzialne za inicjację i weryfikację operacji kartą płatniczą.
* **Printer (prt)** – urządzenie drukujące papierowy bilet.

Komponenty są połączone portami zdarzeń i danych w topologii „gwiazda”: ui → tp →pay→ tp→ prt → tp. Model uwzględnia podstawowe przepływy zdarzeń („wybór biletu”, „polecenie płatności”, „status płatności”, „polecenie druku”, „status druku”) i stanowi bazę do dalszej analizy opóźnień (latency) oraz budżetu zasobów (CPU, pamięć, magistrala).

**2.2. Opis z perspektywy użytkownika**

1. **Wybór biletu**
   * Użytkownik na panelu dotykowym wybiera trasę i typ biletu. Interfejs wysyła sygnał sel\_out do procesora.
2. **Płatność**
   * Procesor, po otrzymaniu żądania, wysyła komendę pay\_cmd\_out do modułu kart (pay\_cmd\_in).
   * Po autoryzacji karta zwraca sygnał pay\_stat\_out, który trafia do procesora jako pay\_stat\_in.
3. **Drukowanie**
   * Jeżeli płatność zakończy się sukcesem, procesor wysyła sygnał prt\_cmd\_out do drukarki (prt\_cmd\_in).
   * Drukarka, po wydaniu biletu, zgłasza prt\_stat\_out powrotnie do procesora (prt\_stat\_in).
4. **Obsługa błędów**
   * W przypadku odrzucenia karty lub braku papieru w drukarce, procesor generuje odpowiedni komunikat zwrotny przez port msg\_in i wyświetla go w interfejsie.

Cały cykl trwa od momentu wyboru biletu do wydruku i sygnalizacji zakończenia operacji.

**3. Spis komponentów AADL z komentarzem**

| **Komponent** | **Typ** | **Interfejs (porty)** | **Opis działania** |
| --- | --- | --- | --- |
| **ui : User\_Interface** | system | - **sel\_out** : out event data port – wysyła sygnał wybranego biletu- **msg\_in** : in event data port – odbiera komunikaty zwrotne | Ekran dotykowy wraz z logiką prezentowania dostępnych opcji i wyświetlania komunikatów o stanie systemu |
| **tp : Ticket\_Processor** | process | - **ui\_req\_in** : in event data port – odbiera wybór z interfejsu użytkownika- **pay\_cmd\_out** : out event data port – inicjuje płatność- **pay\_stat\_in** : in event data port – odbiera status płatności- **prt\_cmd\_out** : out event data port – inicjuje druk- **prt\_stat\_in** : in event data port – odbiera status druku | Centralny proces sterujący kolejnością operacji: przyjmuje żądania od UI, kolejno uruchamia moduły płatności i druku oraz przetwarza ich odpowiedzi |
| **pay : Payment\_Unit** | device | - **pay\_cmd\_in** : in event data port – odbiera polecenie autoryzacji płatności- **pay\_stat\_out** : out event data port – wysyła wynik autoryzacji | Czytnik kart płatniczych, odpowiadający za komunikację z systemem bankowym i raportowanie wyniku |
| **prt : Printer** | device | - **prt\_cmd\_in** : in event data port – odbiera polecenie wydruku biletu- **prt\_stat\_out** : out event data port – wysyła status wydruku | Drukarka termiczna generująca papierowy bilet oraz raportująca zakończenie lub błąd operacji druku |
| **Ticket\_Machine** | system | – bez własnych portów – | System nadrzędny zawierający wszystkie subkomponenty i definiujący ich wzajemne połączenia |

4. Model – rysunek

Obraz zawierający tekst, diagram, zrzut ekranu, linia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rys. 1. Diagram instancji Ticket Machine

Obraz zawierający zrzut ekranu, komputer, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rys. 2. Raport spójności portów – 0 niez 1

1. **Proponowane metody analizy modelu w OSATE i wyniki**

**Check Model (sprawdzanie spójności składniowej)**

* 1. **Opis:** Wbudowana w OSATE weryfikacja poprawności AADL (Xtext Check, Semantic Checks).
  2. **Wynik:** „No problems found” – brak błędów składniowych, wszystkie referencje do typów i portów rozwiązane poprawnie.

**Flow Consistency Analysis**

* 1. **Opis:** Sprawdzenie zgodności definicji przepływów danych (FlowSpecification) i ich odwzorowania na rzeczywiste połączenia portów między komponentami.
  2. **Wynik:** Brak cyklicznych zależności ani niespójności w przepływach; każdy zadeklarowany flow ma odpowiadające mu połączenie c1…c5.

**Port Connection Consistency**

* 1. **Opis:** Weryfikacja, czy kierunki portów (in/out) są zgodne z kierunkiem połączeń oraz czy typy portów są kompatybilne.
  2. **Wynik:** Wszystkie połączenia ui→tp, tp→pay, pay→tp, tp→prt, prt→tp są poprawne – brak ostrzeżeń.

**Latency Analysis (analiza opóźnień)**

* 1. **Opis:** Przy użyciu wtyczki Cheddar lub wbudowanych w OSATE narzędzi timingowych można przydzielić czasy wykonania poszczególnych komponentów i obliczyć maksymalne opóźnienie od wejścia do wyjścia.
  2. **Wynik (symulacja wstępna):** Bez realnych wartości czasów aktywacji i priorytetów wszystkie komponenty traktowane są jako natychmiastowe, więc opóźnienie wynosi 0 ms. Po dodaniu właściwości Compute\_Execution\_Time i harmonogramu RR--FIFO (np. WiPoc) uzyskamy rzeczywiste wartości.

**Resource Budget Analysis (analiza obciążenia zasobów)**

* 1. **Opis:** Ocena procentowego zużycia CPU i pamięci przez każdy subkomponent (współpraca z narzędziem Cheddar lub OSATE Budget).
  2. **Wynik (domyślne ustawienia bez obciążeń):** 0 % użycia CPU i pamięci – model wymaga nadania właściwości zasobowych (m.in. WCET), aby uzyskać realistyczne wyniki.

**Safety / Error Propagation Checks**

* 1. **Opis:** (Opcjonalnie) analiza bezpieczeństwa przepływów błędów – czy sygnał błędu (np. msg\_in z UI) trafia do wszystkich zainteresowanych komponentów i jak jest propagowany.
  2. **Wynik:** Przy braku formalnie zdefiniowanych flow dla sygnałów awarii nie wykonano tej analizy; można ją dołożyć, definiując dodatkowe porty „error” i flow.