Wydział Informatyki i Telekomunikacji
Politechnika Wrocławska

Aplikacja do rozwiązywania nonogramów

Mateusz Wałejko
Nr indeksu: 250335

Praca magisterska napisana
pod kierunkiem
dr. Marcina Michalskiego

Politechnika
Wrocławska

Wrocław 2021

Spis treści

Wstęp

Praca dyplomowa inżynierska jest dokumentem opisującym zrealizowany system techniczny. Praca powinna być napisana poprawnym językiem odzwierciedlającym aspekty techniczne (informatyczne) omawianego zagadnienia. Praca powinna być napisana w trybie bezosobowym (w szczególności należy unikać trybu pierwszej osoby liczby pojedynczej i mnogiej). Zdania opisujące konstrukcję systemu informatycznego powinny być tworzone w stronie biernej. W poniższym dokumencie przykłady sformułowań oznaczono kolorem niebieskim. W opisie elementów systemu, komponentów, elementów kodów źródłowych, nazw plików, wejść i wyjść konsoli należy stosować czcionkę stałej szerokości, np: zmienna `wynik` przyjmuje wartość zwracaną przez funkcję `dodaj(a,b)`, dla argumentów `a` oraz `b` przekazywanych ....

Układ poniższego dokumentu przedstawia wymaganą strukturę pracy, z rozdziałami zawierającymi analizę zagadnienia, opis projektu systemu oraz implementację (dobór podrozdziałów jest przykładowy i należy go dostosować do własnej tematyki pracy).

Wstęp pracy (nie numerowany) powinien składać się z czterech części (które nie są wydzielane jako osobne podrozdziały): zakresu pracy, celu, analizy i porównania istniejących rozwiązań oraz przeglądu literatury, oraz opisu zawartości pracy.

Każdy rozdział powinien rozpoczynać się od akapitu wprowadzającego, w którym zostaje w skrócie omówiona zawartość tego rozdziału.

Praca poświęcona jest wielowarstwowym rozproszonym systemom informatycznym typu „B2B", wspierającym wymianę danych pomiędzy przedsiębiorstwami. Systemy tego typu, konstruowane dla dużych korporacji, charakteryzują się złożoną strukturą poziomą i pionową, w której dokumenty ...

Celem zrealizowanego w ramach pracy projektu było zaprojektowanie i wykonanie aplikacji o następujących założeniach funkcjonalnych:

- wspieranie zarządzania obiegiem dokumentów wewnątrz korporacji z uwzględnieniem ...,
- wspieranie zarządzania zasobami ludzkimi z uwzględnieniem modułów kadrowych oraz ...,
- gotowość do uzyskania certyfikatu ISO ...,
- ....

Istnieje szereg aplikacji o zbliżonej funkcjonalności: ..., przy czym ....

Praca składa się z czterech rozdziałów. W rozdziale pierwszym omówiono strukturę przedsiębiorstwa ..., scharakteryzowano grupy użytkowników oraz przedstawiono procedury związane z obiegiem dokumentów. Szczegółowo opisano mechanizmy .... Przedstawiono uwarunkowania prawne udostępniania informacji ..., w ramach ....

W rozdziale drugim przedstawiono szczegółowy projekt systemy w notacji UML. Wykorzystano diagramy .... Zawarto w niej w pseudokod algorytmów generowania oraz omówiono jego właściwości. ....

W rozdziale trzecim opisano technologie implementacji projektu: wybrany język programowania, biblioteki, system zarządzania bazą danych, itp. Przedstawiono dokumentację techniczną kodów źródłowych interfejsów poszczególnych modułów systemu. Przedstawiono sygnatury metod publicznych oraz ....

W rozdziale czwartym przedstawiono sposób instalacji i wdrożenia systemu w środowisku docelowym.

Końcowy rozdział zawiera podsumowanie uzyskanych wyników.

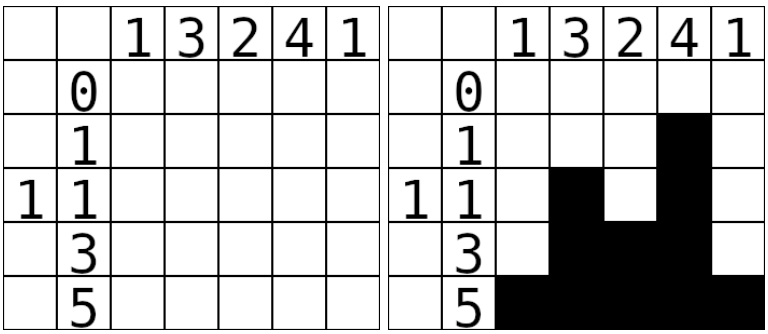
Analiza problemu

W tym rozdziale przedstawiona jest definicja nonogramów oraz sposób ich rozwiązywania. W kolejnej części rozdziału wypisane są także definicje potrzebne do zrozumienia złożoności problemu, jakim jest rozwiązywanie nonogramów.

2.1 Przedstawienie nonogramów

2.1.1 Opis

Nonogramy (znane także jako *Paint by Number* oraz *Picross*) to łamigłówki, w których celem jest odpowiednie wypełnienie komórek na siatce tak, by uzyskać określony wzór (np. obrazek). W tym celu należy kolorować pola zgodnie ze wskazówkami umieszczonymi obok każdego wiersza oraz kolumny na siatce. Wskazówki mają postać ciągu liczb - każda z liczb oznacza ilość wypełnionych komórek z rzędu, a pomiędzy grupami wypełnionych komórek znajduje się przynajmniej jedna pusta komórka.
Uściślając powyższą, nieformalną definicję, nonogram to łamigłówka na siatce wielkości $w \times h$, gdzie w oznacza szerokość planszy wyrażoną w ilości komórek, a h wysokość planszy wyrażoną w ilości komórek. Dla każdego wiersza i kolumny mamy przedstawiony ciąg liczb H_n będący wskazówką dla danej linii. Dany element h_i opisuje blok stworzony z h_i wypełnionych komórek z rzędu, i jeśli h_i nie jest pierwszym elementem ciągu, to blok opisany przez h_i jest oddzielony przynajmniej jedną pustą komórką od bloku opisanego przez h_{i-1}, oraz jeśli h_i nie jest ostatnim elementem ciągu, to blok opisany przez h_i jest oddzielony przynajmniej jedną pustą komórką od bloku opisanego przez h_{i+1}. Linię spełniającą zadaną wskazówkę opisuje wyrażenie regularne: $l = \hat{}0^*1^{h_1}0^+1^{h_2}0^+ \dots 0^+1^{h_n}0^*\$$, gdzie 0, 1 oznaczają odpowiednio pustą i wypełnioną komórkę, h_i jest i-tym elementem ciągu H_n, będącego wskazówką dla wiersza/kolumny, a $|l| = n$, $n = w$ dla wiersza i $n = h$ dla kolumny. Rozwiązaniem nonogramu jest wypełnienie zadanej planszy w taki sposób, by dla każdego wiersza oraz każdej kolumny wskazówki dla nich były spełnione.



(a) przed rozwiązaniem (b) po rozwiązaniu

Rysunek 2.1: Przykładowa plansza

2.1.2 Definicje

Definicja 2.1 *Wskazówką nazywamy ciąg H_i liczb, opisujący ułożenie wypełnionych komórek w danej linii.*

Uwaga. Dla uproszczenia opisu algorytmów następuje założenie, że pusta linia jest opisywana przez wskazówkę będącą ciągiem pustym.

Definicja 2.2 *Instancją problemu nonogramu jest czwórka $N = (w, h, R_n, C_n)$, gdzie w, h to odpowiednio szerokość i wysokość planszy, a R_n i C_n są ciągami wskazówek dla wierszy oraz kolumn.*

2.2 Wymagane zagadnienia matematyczne

2.2.1 Problem decyzyjny

Definicja 2.3 *Problem decyzyjny to problem, na który odpowiedź stanowi 'tak' lub 'nie'.*

Problem decyzyjny to pojęcie kluczowe dla klasyfikacji problemu do wybranej klasy złożoności. By móc zaklasyfikować wybrany problem (np. rozwiązanie nonogramu) do jakiejś klasy złożoności, należy przedstawić go w postaci problemu decyzyjnego.

Przykład 2.1 *Czy zadany nonogram $N = (w, h, R_n, C_n)$ ma rozwiązanie?*

2.2.2 Klasa złożoności P

Definicja 2.4 *Klasa złożoności P zawiera wszystkie problemy decyzyjne, których rozwiązanie można znaleźć w czasie wielomianowym.*

Przykład 2.2 *Znalezienie najkrótszej ścieżki między dwoma punktami w grafie należy do klasy P, ponieważ algorytm Dijkstry znajduje najkrótszą ścieżkę w czasie wielomianowym. Sformułowanie tego problemu w postaci problemu decyzyjnego mogłoby brzmieć następująco: Czy dla danego wejściowego grafu $G = (V, E)$ istnieje ścieżka z punktu $v_1 \in V$ do punktu $v_2 \in V$ o długości niewiększej niż x?*

2.2.3 Klasa złożoności NP

Definicja 2.5 *Klasa złożoności NP zawiera wszystkie problemy decyzyjne, których rozwiązanie dla odpowiedzi pozytywnej można zweryfikować w czasie wielomianowym.*

Przykład 2.3 *Mając zbiór I przedmiotów, gdzie przedmiot i_n to dwójka (v_n, w_n), gdzie v_n to wartość, a w_n to waga, oraz ograniczenie górne na sumę wag wybranych przedmiotów w_{max}, czy można wybrać przedmioty w taki sposób by nie przekroczyć limitu wagi w_{max}, a by suma wartości wybranych przedmiotów była większa lub równa c?*
Tak zadany problem to wersja decyzyjna problemu plecakowego. Być może nie istnieje algorytm znajdujący przydział przedmiotów w czasie wielomianowym, ale mając przedstawione rozwiązanie $S \subseteq I$ można zsumować wartości przedmiotów z S i sprawdzić czy jest to poprawne rozwiązanie.

Należy zauważyć, że każdy problem z klasy P należy także do klasy NP, ponieważ rozwiązanie problemu decyzyjnego jest jednym ze sposobów weryfikacji poprawności jego rozwiązania. To czy $P = NP$ jest jak do tej pory nierozwiązanym problemem.

2.2.4 Redukcja wielomianowa

Definicja 2.6 *Problem A jest redukowalny do problemu B w czasie wielomianowym, jeśli wejścia dla problemu A można przekształcić na wejścia dla problemu B w czasie wielomianowym, a następnie rozwiązać problem A wywołując procedurę rozwiązującą problem B wielomianową ilość razy.*

Przykład 2.4 *Można zdefiniować mnożenie liczb $a \cdot b$ za pomocą operacji dodawania w następujący sposób:*

$$a \cdot b = \underbrace{a + a + \ldots + a}_{b}$$

Należy zauważyć, że jeśli problem A jest redukowalny do problemu B w czasie wielomianowym, a problem B należy do klasy P, to także problem A należy do klasy P, jako że sposobem na jego rozwiązanie jest użycie redukcji wielomianowej by traktować go jako instancję problemu B, a następnie rozwiązanie go za pomocą algorytmu działającego w czasie wielomianowym.

Wniosek 2.1 *Jeśli istnieje redukcja z A w B w czasie wielomianowym, to B jest co najmniej tak złożony jak A.*

2.2.5 Klasa problemów NP-trudnych

Definicja 2.7 *Problem H należy do klasy problemów NP-trudnych, jeśli każdy problem w klasie NP jest redukowalny do H w czasie wielomianowym.*

W przypadku klasy problemów NP-trudnych nie ma wymogu, by należące do niej problemy były problemami decyzyjnymi.

Przykład 2.5 *Przykładem problemu NP-trudnego jest problem spełnialności (SAT): 'Czy dla danej formuły logicznej istnieje wartościowanie, dla którego zadana formuła jest spełniona?'. Przynależność tego problemu do tej klasy została udowodniona w 1971 roku przez Stephena Cooka i Leonida Levina w dowodzie twierdzenia Cooka-Levina [3].*

Dla udowadniania przynależności problemu do tej klasy kluczowa jest obserwacja, że istnienie redukcji wielomianowej z A w B implikuje przynależność B do tej klasy, o ile A także do niej należy.

2.2.6 Klasa problemów NP-zupełnych

Definicja 2.8 *Problem decyzyjny C należy do klasy problemów NP-zupełnych, jeśli należy do klas problemów NP-trudnych oraz NP.*

Wniosek 2.2 *Pokazanie, że problem decyzyjny A jest NP-zupełny sprowadza się do pokazania, że istnieje redukcja wielomianowa z problemu H z klasy problemów NP-trudnych, oraz że rozwiązanie problemu A można zweryfikować w czasie wielomianowym.*

2.3 Przypisanie problemu rozwiązania nonogramów do odpowiedniej klasy złożoności

Mając zdefiniowane pojęcia potrzebne do klasyfikacji problemu do odpowiedniej klasy złożoności, należy znaleźć klasę do jakiej należy rozwiązywanie nonogramów. Z uwagi na specyfikę klas, klasyfikacji poddana zostanie decyzyjna wersja problemu, tj. '*Czy zadany nonogram $N = (w, h, R_n, C_n)$ ma rozwiązanie?*'.

2.3.1 Problem rozwiązania nonogramu jest w NP

Niech $M_{h,w}$ będzie macierzą oznaczająca rozwiązanie zadanego nonogramu $N = (w, h, R_n, C_n)$, gdzie $m_{i,j}$ oznacza stan komórki w wierszu i i kolumnie j, oraz $m_{i,j} = 1$, jeśli komórka jest wypełniona, a $m_{i,j} = 0$ jeśli pusta. Macierz $M_{h,w}$ jest mapowana na $h + w$ list, będących ciągami stanów komórek w kolejnych wierszach, i kolumnach.

Do weryfikacji rozwiązania użyjemy następującej procedury:

Pseudokod 2.1: Poprawność rozwiązania w osi

```
Input: Lista linii L, Lista wskazówek LH, długość linii n
Output: Poprawność rozwiązania w osi (true/false)
for i ← 1 to |L| do
    Li ← L[i];
    Hi ← LH[i];
    a ← 0;
    A ← [];
    for c ∈ Li do
        if c = 1 then
            a ← a + 1
        else
            if a > 0 then
                A.push(a);
                a ← 0;
    if a > 0 then
        A.push(a);
    for j ← 1to|Hi| do
        if A[j] ≠ Hi[j] then
            return false;
return true;
```

W procedurze 2.1 następuje weryfikacja rozwiązania w danej osi. Przykładowo, wywołując procedurę 2.1 dla listy wierszy i ich wskazówek, weryfikujemy Poprawność rozwiązania w poziomie. Weryfikacja rozwiązania następuje przez wywołanie procedury dwukrotnie, dla wierszy oraz kolumn. Jeśli w obu przypadkach procedura zwróci `true`, to rozwiązanie jest poprawne.
Czas wykonania procedury jest zależny od wielkości planszy. Zewnętrzna pętla wykonuje się tyle razy, ile jest linii w osi (h w przypadku wierszy, w w przypadku kolumn). Na początku pętli dochodzi do ekstrakcji pewnych danych do lokalnych zmiennych oraz inicjalizacji tablicy - w zależności od jezyka użytego do implementacji, ta grupa operacji zajmuje czas stały bądź liniowy. Następnie uruchamiana jest pierwsza wewnętrzna pętla. W tej pętli analizowane są dane w danej linii, by zmapować układ jej komórek do wskazówki jaką reprezentuje. Złożoność operacji w każdej iteracji jest stała, jeśli założymy że powiększenie tablicy o dodatkowy element wymaga stałego czasu - w p.p. czas wykonania iteracji może być liniowy. Ilość wykonań tej pętli zależy od długości linii. Po wykonaniu pierwszej pętli, w zależności od układu stanu komórek w linii, może dojść do kolejnego powiększenia tablicy o dodatkowy element - złożoność nie przekracza liniowej. Na końcu zewnętrznej pętli wykonywana jest druga pętla, która iteruje po elementach wskazówki zadanej w rozwiązaniu, i porównuje ich watość do analogicznych elementów we wskazówce odtworzonej z układu linii. Rozbieżność oznacza, że rozwiązanie nie jest prawidłowe, i procedura przedwcześnie zakańcza wykonanie. Długość wskazówki można z góry ograniczyć przez $\lceil \frac{x}{2} \rceil$, gdzie x jest długością linii.
Zadana procedura sprawdza poprawność rozwiązania nonogramów, a jej złożoność, w zależności od implementacji operacji na tablicach, może wynosić $\mathcal{O}(n^2)$ bądź $\mathcal{O}(n^3)$. Zaproponowana procedura ma złożoność wielomianową, zatem problem decyzyjny rozwiązywania nonogramów należy do klasy *NP*.

2.3.2 Problem rozwiązania nonogramu jest NP-trudny

Dowód NP-trudności rozwiązywania nonogramów jest obszerny i wykracza poza zakres tej pracy. Przykładowy dowód jest opisany w pracy [4] i jego zarys jest następujący. Autor rozpoczyna dowód od powołania się na NP-trudność gry na grafach, nazwanej jako *Bounded Nondeterministic Constraint Logic*. Następnie, poprzez redukcję, autor udowadnia NP-trudność zmodyfikowanej wersji gry, określonej na grafach planarnych. Po udowodnieniu tego faktu, autor konstruuje redukcję wielomianową z planarnej *Bounded Nondeterministic Constraint Logic* w rozwiązywanie nonogramów, tym samym udowadniając ich przynależność do tej klasy

problemów.

2.3.3 Problem rozwiązania nonogramu jest NP-zupełny

Pokazawszy, że zadany problem jest w NP, oraz jest NP-trudny, pokazane zostało że problem ten jest NP-zupełny.

Projekt systemu

W tym rozdziale przedstawiono szczegółowy projekt systemy w notacji UML uwzględniający wymagania funkcjonalne opisane w rozdziale 2. Do opisu relacji pomiędzy składowymi systemu wykorzystano diagramy .... Przedstawiono w pseudokodzie i omówiono algorytmy generowania ....

3.1 Grupy użytkowników i założenia

Architektura systemu ... jest wielowarstwowa i rozproszona, przy czym .... Podsystem ... jest systemem zbiorczym dla danych ... wysyłanych do serwera ....

Taka architektura jest zgodna z wzorcem projektowym MVC[1] (ang. Model-View-Controller). Przetwarzanie danych odbywa się ....

3.2 Przypadki użycia i scenariusze

W tej sekcji należy przedstawić przypadki użycia oraz odpowiadające im scenariusze dla poszczególnych grup użytkowników ....

3.3 Diagramy klas

W tej sekcji należy przedstawić diagramy klas dla odpowiednich elementów systemu zidentyfikowane na podstawie wcześniejszych rozważań

3.4 Diagramy aktywności

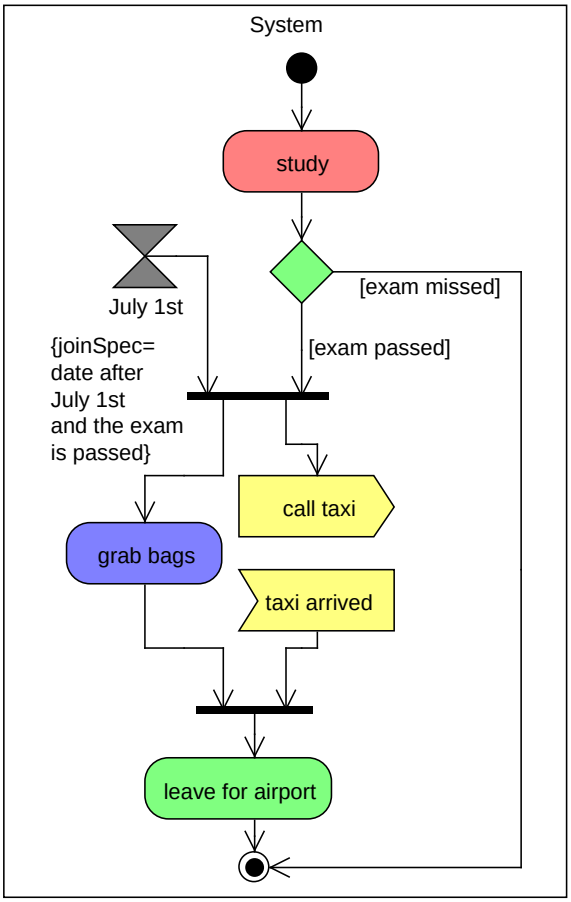
W tej sekcji należy przedstawić diagramy aktywności dla elementów systemu i odpowiednich procesów wynikające z wcześniejszej analizy.

W niniejszym rozdziale przedstawiono diagramy aktywności .... Diagram na rysunku 3.1 przedstawia ....

3.5 Diagramy sekwencji

W tej sekcji należy przedstawić diagramy sekwencji dla obiektów systemu zidentyfikowanych na podstawie wcześniejszych rozważań. Należy wykorzystać nazewnictwo wprowadzone w poprzednich rozdziałach, w szczególności odpowiadające definicjom wprowadzonych klas.

[1] Należy odnieść się do wykorzystywanych wzorców projektowych



Rysunek 3.1: Diagram aktywności związany z procesem rejestracji dokumentu.

3.6 Diagramy stanów

W tej sekcji należy przedstawić diagramy stanów w których może znaleźć się system. Diagramy te są szczególnie istotne przy projektowaniu systemów czasu rzeczywistego.

3.7 Projekt bazy danych

W tej sekcji należy przedstawić projekt bazy danych. Należy omówić wycinek rzeczywistości i odpowiadające mu zidentyfikowane elementy systemu, których wartości będą podlegać utrwalaniu. Należy przedyskutować wybór typów danych dla atrybutów poszczególnych obiektów. Należy uzasadnić wybór platformy DBMS. Dla relacyjnych baz danych należy przedyskutować jej normalizację.

3.8 Opis protokołów

W tej sekcji należy omówić protokoły wykorzystywane przez komponenty systemu. Omówić formaty komunikatów i zilustrować je przykładami.

3.9 Opis algorytmów

W tej sekcji należy wymienić i przedyskutować algorytmy wykorzystywane w systemie. Algorytmy należy przedstawić w pseudokodzie (wykorzystać pakiet `algorithm2e`). Omówienia poszczególnych kroków algorytmów powinny zawierać odwołania do odpowiednich linii pseudokodu. Dla zaproponowanych autorskich algorytmów należy przeprowadzić analizę ich złożoności czasowej i pamięciowej.

Algorytm bąblowania jest przedstawiony w Pseudokodzie 3.1.

Pseudokod 3.1: Wyporność przez bąblowanie

```
   Input: Zbiór bąbli B
   Output: Wyporność W
1  foreach b ∈ B do
2      Process(b);
3      for i ← 1 to |B| do
4          if Calculate(EW(i,b)) ⩽ 0 then
5              b ← 2 ∗ b;
6  while B ≠ ∅ do
7      for j ← 1 to |B| do
8          if Calculate(FT(j,b̂)) ⩽ 0 then
9              w ← 2 ∗ b̂;
10             W ← W ∪ {w};
11             B ← B \ {b};
```

Implementacja systemu

4.1 Opis technologii

Należy tutaj zamieścić krótki opis (z referencjami) do technologii użytych przy implementacji systemu.

Do implementacji systemu użyto języka JAVA w wersji . . . , szczegółowy opis można znaleźć w [1]. Interfejs zaprojektowano w oparciu o HTML5 i CSS3 [2].

4.2 Omówienie kodów źródłowych

Kod źródłowy 4.1 przedstawia opisy poszczególnych metod interfejsu: `WSPodmiotRejestracjaIF`. Kompletne kody źródłowe znajdują się na płycie CD dołączonej do niniejszej pracy w katalogu `Kody` (patrz Dodatek A).

Kod źródłowy 4.1: Interfejs usługi Web Service: `WSPodmiotRejestracjaIF`.

```
package erejestracja.podmiot;
import java.rmi.RemoteException;
// Interfejs web serwisu dotyczącego obsługi podmiotów i rejestracji.
public interface WSPodmiotRejestracjaIF extends java.rmi.Remote{
// Pokazuje informacje o danym podmiocie.
// parametr: nrPeselRegon - numer PESEL podmiotu lub numer REGON firmy.
// return: Podmiot - obiekt transportowy: informacje o danym podmiocie.
public Podmiot pokazPodmiot(long nrPeselRegon) throws RemoteException;
// Dodaje nowy podmiot.
// parametr: nowyPodmiot - obiekt transportowy: informacje o nowym podmiocie.
// return: true - jeśli podmiot dodano, false - jeśli nie dodano.
public boolean dodajPodmiot(Podmiot nowyPodmiot) throws RemoteException;
// Usuwa dany podmiot.
// parametr: nrPeselRegon - numer PESEL osoby fizycznej lub numer REGON firmy.
// return: true - jeśli podmiot usunięto, false - jeśli nie usunięto.
public boolean usunPodmiot(long nrPeselRegon) throws RemoteException;
// Modyfikuje dany podmiot.
// parametr: podmiot - obiekt transportowy: informacje o modyfikowanym podmiocie.
// return: true - jeśli podmiot zmodyfikowano, false - jeśli nie zmodyfikowano.
public boolean modyfikujPodmiot(Podmiot podmiot) throws RemoteException;
// Pokazuje zarejestrowane podmioty na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego - numer dowodu rejestracyjnego.
// return: PodmiotRejestracja[] - tablica obiektów transportowych: informacje o
// wszystkich zarejestrowanych podmiotach.
public PodmiotRejestracja[] pokazZarejestrowanePodmioty(
String nrDowoduRejestracyjnego) throws RemoteException;
// Nowa rejestracja podmiotu na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego - numer dowodu rejestracyjnego.
// parametr: nrPeselRegon - numer PESEL podmiotu lub numer REGON firmy.
// parametr: czyWlasciciel - czy dany podmiot jest właścicielem pojazdu.
// return: true - jeśli zarejestrowano podmiot, false - jeśli nie zarejestrowano.
public boolean zarejestrujNowyPodmiot(String nrDowoduRejestracyjnego,
```

```
long nrPeselRegon, boolean czyWlasciciel) throws RemoteException;
// Usuwa wiązanie pomiędzy danym podmiotem, a dowodem rejestracyjnym.
// parametr: nrDowoduRejestracyjnego - numer dowodu rejestracyjnego.
// parametr: nrPeselRegon - numer PESEL podmiotu lub numer REGON firmy.
// return: true - jeśli podmiot wyrejestrowano, false - jeśli nie wyrejestrowano.
public boolean wyrejestrujPodmiot(String nrDowoduRejestracyjnego,
long nrPeselRegon) throws RemoteException;
```

Kod źródłowy 4.2 przedstawia procedurę przetwarzającą żądanie. Hasz utrwalany `%granulacja` wykorzystywany jest do komunikacji międzyprocesowej.

Kod źródłowy 4.2: Przetwarzanie żądania - procedura `process_req()`.

```
sub process_req(){
  my($r) = @_;
  $wyn = "";
  if ($r=~/get/i) {
        @reqest = split("_",$r);
        $zad = $reqest[0];
        $ts1 = $reqest[1];
        $ts2 = $reqest[2];
        @date1 = split(/\D/,$ts1);
        @date2 = split(/\D/,$ts2);
        print "odebralem:_$r";
        $wyn = $wyn."zadanie:_$zad\n";
        $wyn = $wyn."czas_od:_"."$date1[0]"."-"."$date1[1]"."-"."$date1[2]"."_"."$date1[3]".":"."$
        $wyn = $wyn."czas_do:_"."$date2[0]"."-"."$date2[1]"."-"."$date2[2]"."_"."$date2[3]".":"."$
        $wyn = $wyn.&sym_sens($ts1,$ts2);
        return $wyn;
  }
  if ($r=~/set gt/i) {
        @reqest = split("_",$r);
        $zad = $reqest[0];
        $ts1 = $reqest[1];
        $ts2 = $reqest[2];
        $gt = $reqest[2];
        dbmopen(%granulacja,"granulacja_baza",0644);
        $granulacja{"gt"}=$gt;
        dbmclose(%granulacja);
        $wyn = "\'GT\'_zmienione_na:_$gt";
  }
}
```

Instalacja i wdrożenie

W tym rozdziale należy omówić zawartość pakietu instalacyjnego oraz założenia co do środowiska, w którym realizowany system będzie instalowany. Należy przedstawić procedurę instalacji i wdrożenia systemu. Czynności instalacyjne powinny być szczegółowo rozpisane na kroki. Procedura wdrożenia powinna obejmować konfigurację platformy sprzętowej, OS (np. konfiguracje niezbędnych sterowników) oraz konfigurację wdrażanego systemu, m.in. tworzenia niezbędnych kont użytkowników. Procedura instalacji powinna prowadzić od stanu, w którym nie są zainstalowane żadne składniki systemu, do stanu w którym system jest gotowy do pracy i oczekuje na akcje typowego użytkownika.

Podsumowanie

W podsumowanie należy określić stan zakończonych prac projektowych i implementacyjnych. Zaznaczyć, które z zakładanych funkcjonalności systemu udało się zrealizować. Omówić aspekty pielęgnacji systemu w środowisku wdrożeniowym. Wskazać dalsze możliwe kierunki rozwoju systemu, np. dodawanie nowych komponentów realizujących nowe funkcje.

W podsumowaniu należy podkreślić nowatorskie rozwiązania zastosowane w projekcie i implementacji (niebanalne algorytmy, nowe technologie, itp.).

Bibliografia

[1] Java technology. Web pages: http://www.oracle.com/technetwork/java/.

[2] B. Frain. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing, 2012.

[3] L. L. Stephen Cook. The complexity of theorem-proving procedures, 1971.

[4] J. N. van Rijn. Playing games. the complexity of klondike, mahjong, nonograms and animal chess. https://liacs.leidenuniv.nl/assets/2012-01JanvanRijn.pdf, 2012.

Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.