Návrh softvéru
Smelý zajko sprievodcom

Tvorba informačných systémov

Autori: Natália Ďurisová, Dominik Knechta,
Martin Mudroch, Miroslav Gregorec

Obsah

1 Úvod

Cieľom tohto dokumentu je špecifikovať návrh softvéru k projektu Smelý zajko sprievodcom

Dokument je štruktúrovaný do troch logických častí, ktoré sú:

- Konceptuálna analýza
  Cieľom konceptuálnej analýzy je podľa katalógu požiadaviek schváleného zadávateľom projektu - analyzovať používateľov systému, prostredníctvom diagramov prezentovať funkcionality systému (robota) a predviesť prvotný návrh užívateľského prostredia.
- Analýza technológií, dekompozícia a dátový model
  Cieľom tohto dokumentu je analyzovať a popísať použité technológie, ktoré budú použité pri tvorbe projektu smelý zajko sprievodcom. Dané technológie budú v dokumente popísané slovne a taktiež znázornené pomocou komponentového diagramu. V dokumente sú okrem technológií na správu robota, popísané aj technológie použité pre webové prostredie a vzájomnú komunikáciu medzi robotom a webom.
- Návrh

2 Konceptuálna analýza

2.1 Analýza používateľov

Používateľov tohto projektu možno rozdeliť na dve skupiny :

- Používatelia webovej stránky
- Používatelia robota

Používatelia webovej stránky:

1) **Anonymní používatelia**
   sú používatelia, ktorí získajú ľubovoľným spôsobom odkaz na stránku. Títo budú mať obmedzené práva – nebudú môcť ovládať robota. Cieľom anonymných používateľov je oboznámiť sa na stránke s projektom, prezerať si históriu využitia robota a zistiť či sa aktuálne robot používa.
2) **Admin**
   Administrátor bude manuálne vložený do systému, pričom po prihlásení bude mať oproti anonymnému používateľovi možnosť zapnúť a vypnúť robota na diaľku či zrušiť aktuálnu navigáciu.

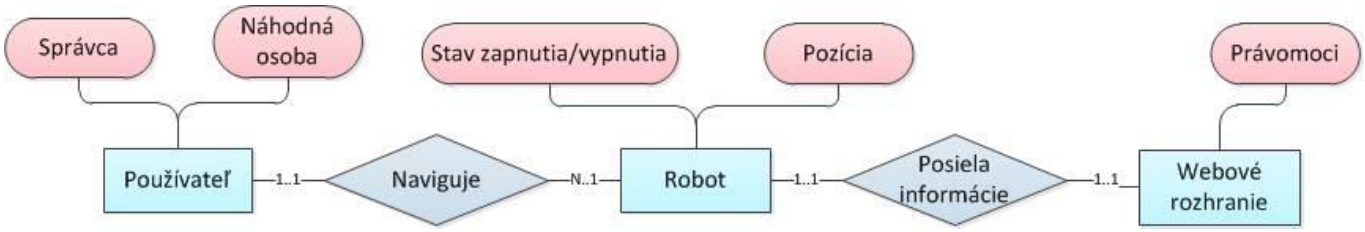
Používatelia robota :

1) **Stretnutá osoba**
   Je používateľ, ktorý stretne robota pri fáze prechádzania a následne po vyzvaní robotom využije možnosť navigácie po zadaní miestnosti na numerickej klávesnici.
2) **Správca**
   Je osoba, ktorá zapne robota a uvedie ho do chodu vo svojom pavilóne.

Jednotlivé kategórie používateľov webovej stránky spolu s akciami, ktoré môžu vykonávať, je možné názorne vidieť v kategórii Diagramy, časť use-case diagram.

2.2 Diagramy

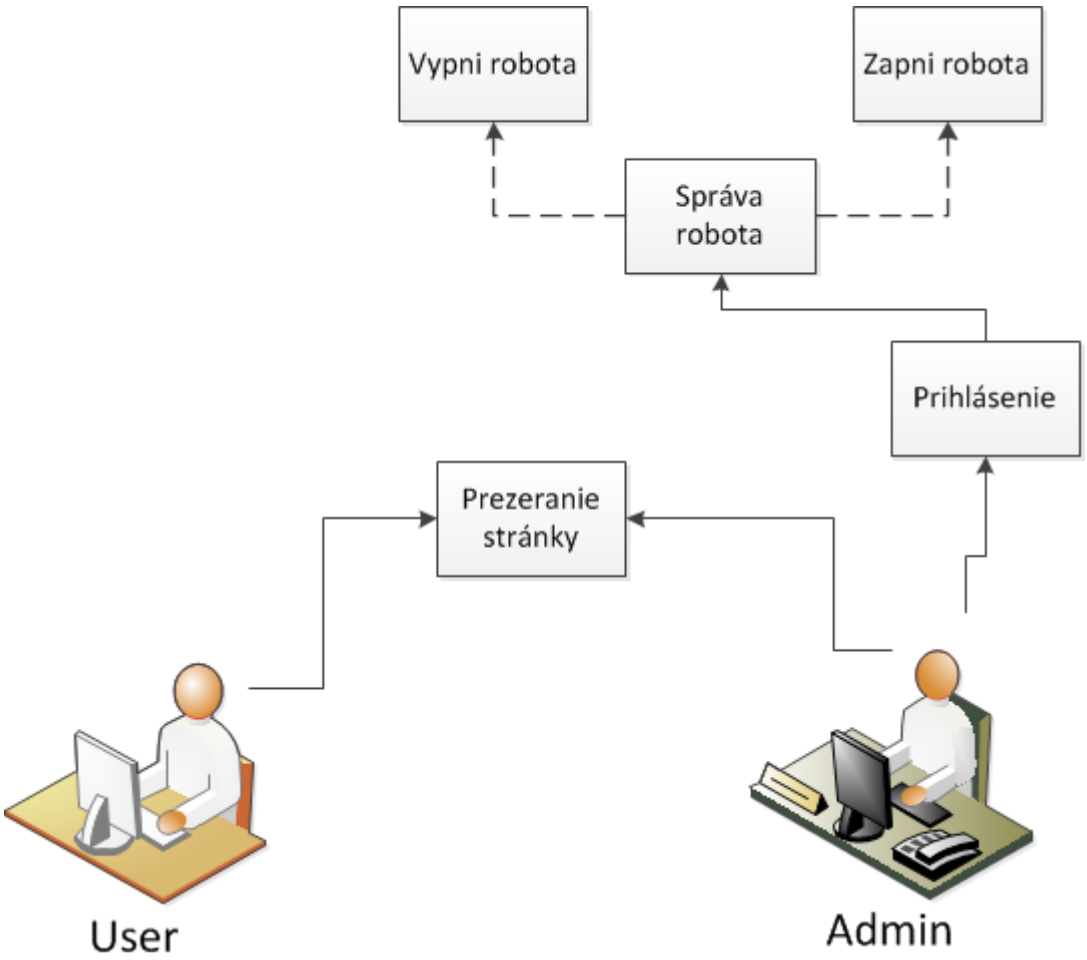
2.2.1 Entitno-relačný diagram



Obrázok 1 Entitno-relačný diagram

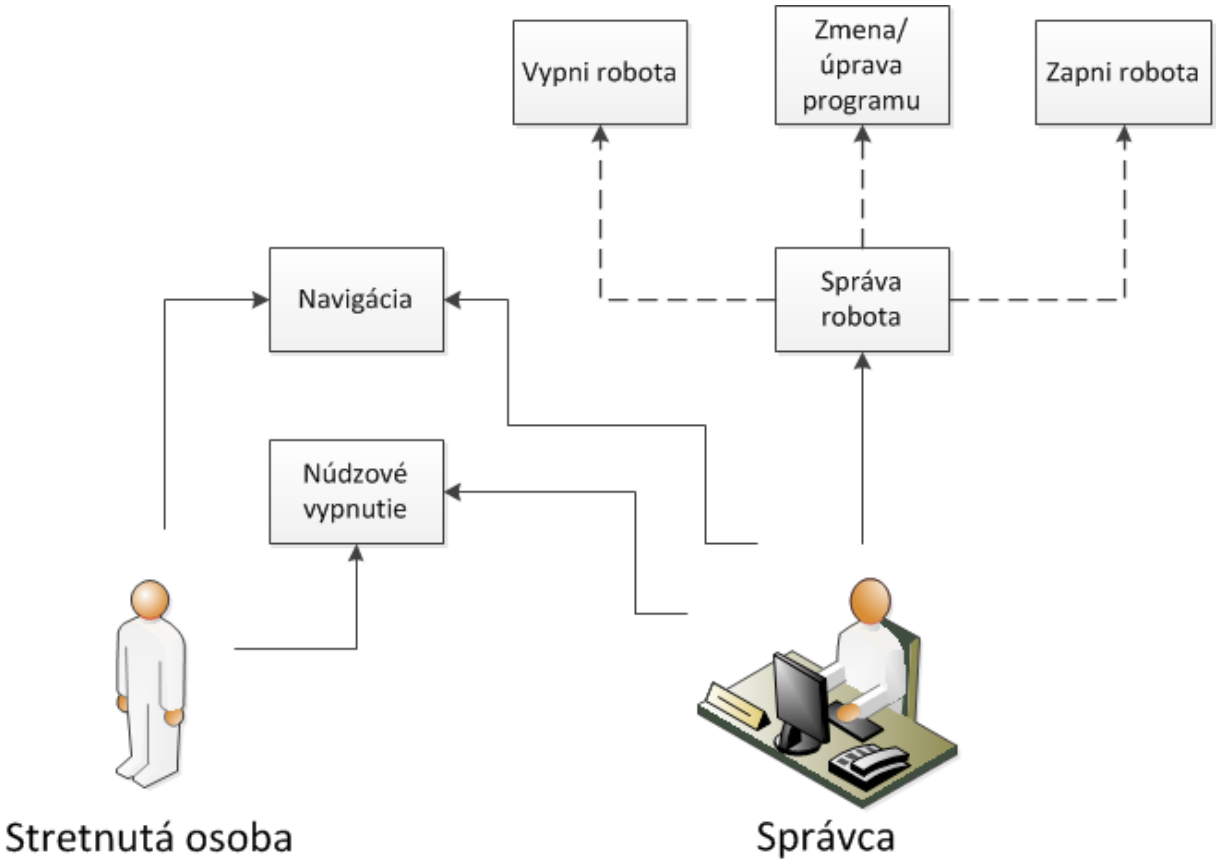
2.2.2 Use-case diagram

2.2.2.1 Diagram pre používateľov webového rozhrania



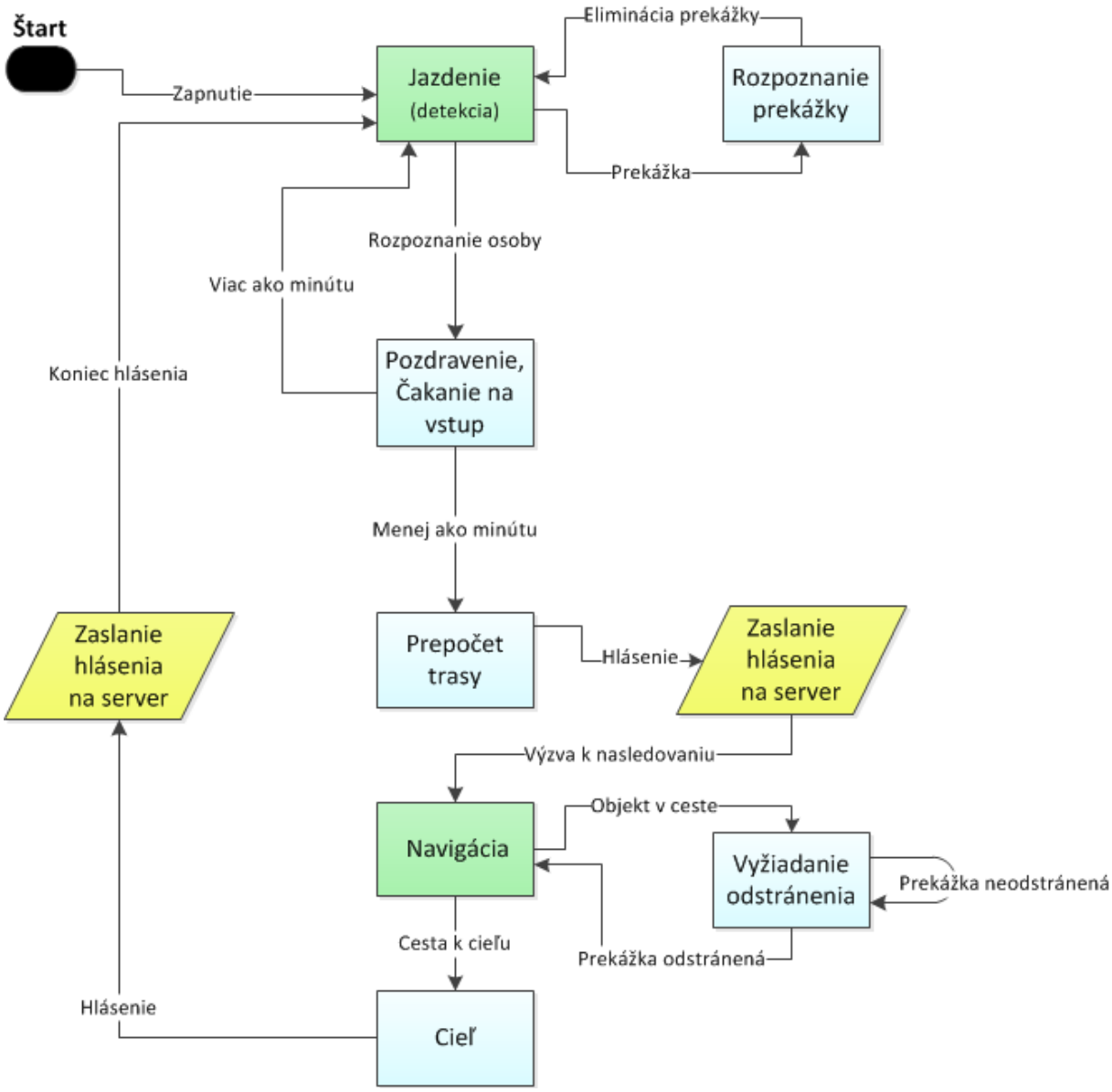
Obrázok 2 Use Case diagram pre webové rozhranie

2.2.2.2 Diagram pre používateľov robota



Obrázok 3 Use Case diagram pre robota

2.2.3 Stavový diagram



Obrázok 4 Stavový diagram robota

2.3 Používateľské rozhranie

V tejto časti sa venujeme predbežnému náčrtu našej webovej stránky, na ktorej si môže používateľ pozrieť aktuálny stav robota a jeho predchádzajúce splnené úlohy.

Webová stránka bude mať dva rozhrania, pre používateľa a admina, ktoré budú takmer podobné s výnimkou, že v administrátorskom prostredí bude možnosť spravovať robota.

2.3.1 Spoločné rozhranie (Používateľský režim)

Naše rozhranie bude jednoduchá webová stránka, ktorá bude pozostávať z názvu a popisu projektu, ďalej sa tu bude nachádzať informácia o stave robota (či je zapnutý alebo vypnutý a či sa prechádza alebo práve naviguje). Pod momentálnym stavom robota bude zobrazená história spracovaných úloh. História spracovaných úloh bude spracovaná v tabuľke, ktorá bude vyzerať približne ako na znázornenej ukážke.

Stavy robota:

Obrázok 5 Grafické znázornenie stavu robota

História spracovaných úloh:

Začiatok navigácie	Koniec navigácie	Štart navigácie	Cieľ navigácie	Celkový čas navigácie
09:50:00	11:40:23	I-11	I-9	15min
12:20:03	12:24:25	I-11	I-24	20 min
12:20:03	12:24:25	I-11	I-24	06 min
13:50:16	15:24:25	I-11	I-24	18 min
21:20:03	21:24:25	I-11	I-24	9 min
22:20:03	23:24:25	I-11	I-24	04 min

Obrázok 6 Grafické znázornenie tabuľky spracovaných úloh

2.3.2 Administrátorský režim

Na stránke bude okrem hore spomínaných komponentov taktiež tlačidlo „Administrácia prostredia“. Po kliknutí na toto tlačidlo, bude užívateľ vyzvaný zadať heslo, ktoré ho presmeruje na stránku určenú pre administrátorov. Heslo budú poznať len vopred určení správcovia robota. Administrátorský režim stránky vyzerá veľmi podobne ako používateľský, avšak v tomto režime má admin právo vypnúť a zapnúť robota na diaľku a taktiež zrušiť robotovi momentálne zadanú úlohu.

3 Analýza technológií, dekompozícia a dátový model

3.1 Analýza technológií

Technológie môžeme rozdeliť na 2 skupiny podľa využitia:

- Technológie na správu robota
- Technológie pre webové rozhranie

3.1.1 Technológie na správu robota:

1. **Programovací jazyk (C++)**
   Podstatou celého projektu je programovací jazyk, v ktorom budú naprogramované všetky metódy na riadenie robota. Keďže úlohou robota je aj rozpoznávať postavy rozhodli sme pre jazyk C++, pretože najlepšie spolupracuje s knižnicou OpenCV bližšie opísanou nižšie. C++ je viacparadigmový programovací jazyk vyššej úrovne na všeobecné použitie, ktorý umožňuje pracovať aj s prostriedkami nízkej úrovne. Má statickú typovú kontrolu, podporuje procedurálne programovanie, dátovú abstrakciu, objektovo orientované programovanie, ale aj generické programovanie.

2. **Knižnica na rozpoznávanie postáv (OpenCV)**
   Počas jazdenia robot sníma prostredie pomocou webkamery. Obraz z webkamery sa odosiela na rozpoznanie či sa na ňom nachádza nejaká postava. Takúto úlohu dokáže spracovať vybratá knižnica OpenCV. OpenCV je tzv. open-source, multiplatformová knižnica určená pre manipuláciu s obrazom. Je zameraná predovšetkým na počítačové videnie a spracovanie obrazu v reálnom čase.

3. **Mapy vo formáte SVG**
   Nato aby robot mohol správne navigovať po informatickom pavilóne potrebuje poznať polohu miestností a taktiež kde sa práve nachádza. Na tento účel použijeme už vopred pripravené SVG mapy informatického pavilónu, v ktorých sú zaznamenané súradnice potrebných miestností. Scalable Vector Graphics (SVG) je značkovací jazyk z rodiny značkovacích jazykov XML, ktorý je určený na opis dvojrozmernej, statickej alebo animovanej vektorovej grafiky.

4. **Hlasový syntetizátor pre komunikáciu s človekom**
   Keďže robot komunikuje s človekom aj verbálne, potrebujeme na daný problém takzvaný TTS (text-to-speech) program. Robot komunikuje s človekom slovenským jazykom, preto bude potrebné použiť syntetizátor, ktorý podporuje daný jazyk.

3.1.2 Technológie pre webové rozhranie :

1. **Hypertext Markup Language (HTML)**
   HTML je jednoduchý značkovací jazyk bežne používaný na tvorbu statických webstránok zobrazovaných vo webovom prehliadači. Tvorí základ pre rozšírenie stránok o dynamické prvky pomocou jazykov ako napr. PHP. Jeho výhodou je jednoduchosť a bežná podpora zo strany internetových prehliadačov. Veľmi často je používaný spolu s kaskádovými štýlmi (CSS). Jazyk bude použitý pre statický návrh webové prostredia k robotovi, ktorý ma informačný aj spravovací charakter.

2. **Cascading Style Sheets (CSS)**
   Kaskádové štýly alebo CSS je všeobecné rozšírenie (X)HTML. Organizácia W3C označuje CSS ako jednoduchý mechanizmus na vizuálne formátovanie internetových dokumentov. Štýly umožňujú oddeliť štruktúru HTML alebo XHTML od vzhľadu. Pre náš projekt bude CSS použité na design webovej stránky.

3. **Hypertext Preprocessor (PHP)**
   PHP je populárny open source skriptovací jazyk, ktorý sa používa najmä na programovanie klient-server aplikácií (na strane servera) a pre vývoj dynamických webových stránok**.** Pre náš projekt bude PHP využitý na prijímanie údajov o stavoch robota, ich následné spracovanie a taktiež na prihlasovanie do administrátorského prostredia.

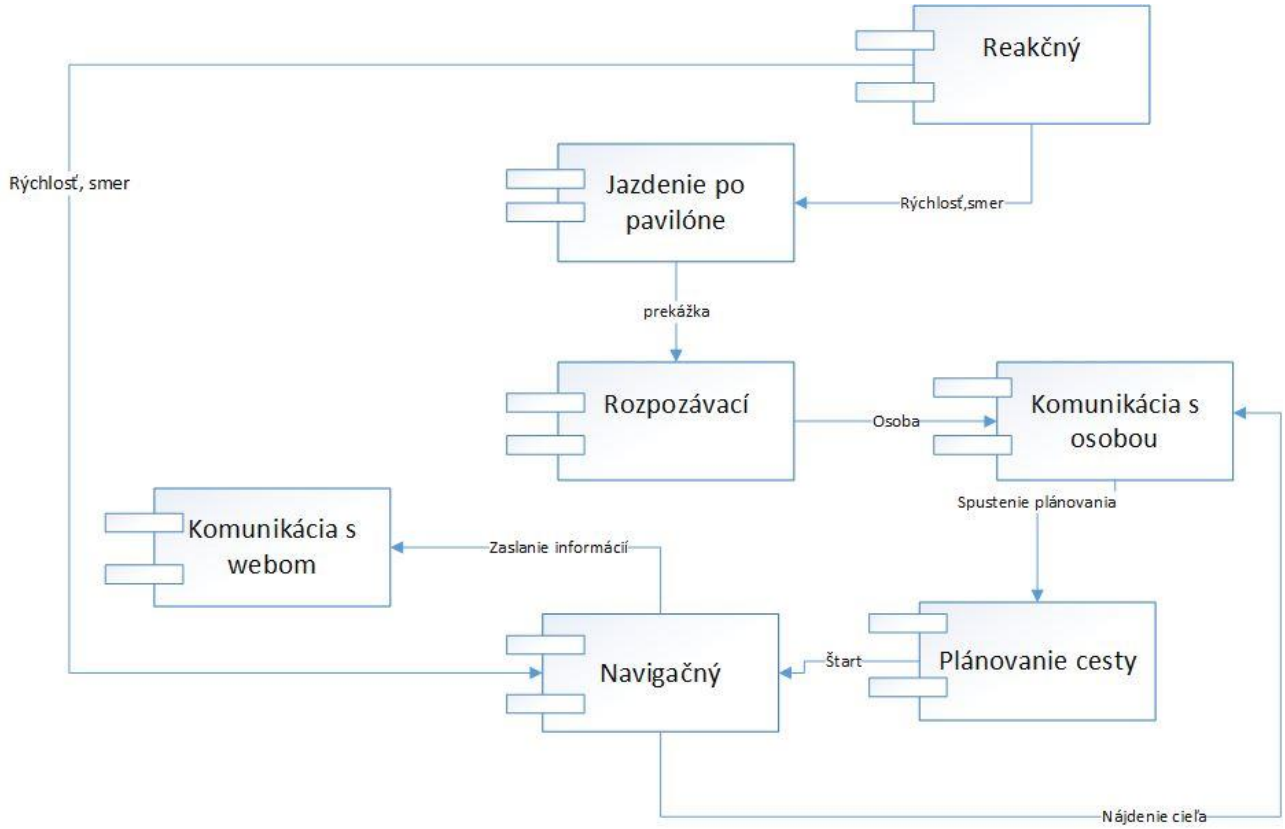
3.2 Dekompozícia

Projekt bude rozdelený na dve časti:

1. Program pre robota.
   Tento program bude obsahovať triedy potrebné pre jazdu robota po pavilóne, komunikáciu s človekom, komunikáciu so serverom a vyhľadávanie miestností.
   - Jazda:
     Má metódy na korigovanie smeru a rýchlosti jazdy, výpočet aktuálnej pozície, kontrolu výskytu osoby alebo prekážky. Pamätá si svoju aktuálnu pozíciu a cieľ jazdy.
   - Rozpoznávanie prítomnosti človeka:
     Pomocou OpenCV sa spracúva výstup z kamery a reaguje na prítomnosť človeka.
   - Komunikácia s človekom:
     Obsahuje metódy na hlasový výstup robota, spracovanie a kontrolu vstupu a časovač.
   - Komunikácia so serverom:
     Metódy na odosielanie dát na server a na prijímanie dát zo serveru.

2. Webová stránka.
   Na webovej stránke budú bežnému užívateľovi k dispozícii údaje o momentálnom stave robota a história jeho úloh. Admin má možnosť robota vypnúť alebo zapnúť.

3.2.1 Komponentový diagram



Obrázok 7

3.2.2 Popis komponentov

3.2.2.1 Reakčný komponent

Komponent je potrebný pre samotný štart robota, po spustení sa zapnú senzory kamera a spustí sa aj samotný program prechádzania robota. V reakčnom komponente sa ďalej sleduje vzdialenosť robota od stien a zaisťuje sa správny pohyb po chodbe.

3.2.2.2 Komponent Jazdenie po pavilóne

Po spustení robota sa zapína program, ktorý zabezpečuje samotnú jazdu po pavilóne, čo zahrňuje pamätanie pozície robota, ako aj kontrolovanie prekážok pri robotovi.

3.2.2.3 Rozpoznávací komponent

Tento komponent využíva predovšetkým knižnicu OpenCV, pomocou ktorej sa snaží rozpoznať objekt nachádzajúci sa pred robotom. Po zistení, že pred robotom sa nachádza osoba, komponent odošle informáciu a spustí sa komunikačný komponent .

3.2.2.4 Komponent komunikácia s osobou

V tomto komponente sa prevažne využíva hlasový syntetizátor, ktorý oboznámi používateľa o postupe navigácie.

3.2.2.5 Komponent plánovanie cesty

Po zadaní vstupu od užívateľa sa v tomto komponente vypočíta najkratšia cesta k zvolenému miestu a následne sa spustí komponent navigácie.

3.2.2.6 Komponent Navigácia

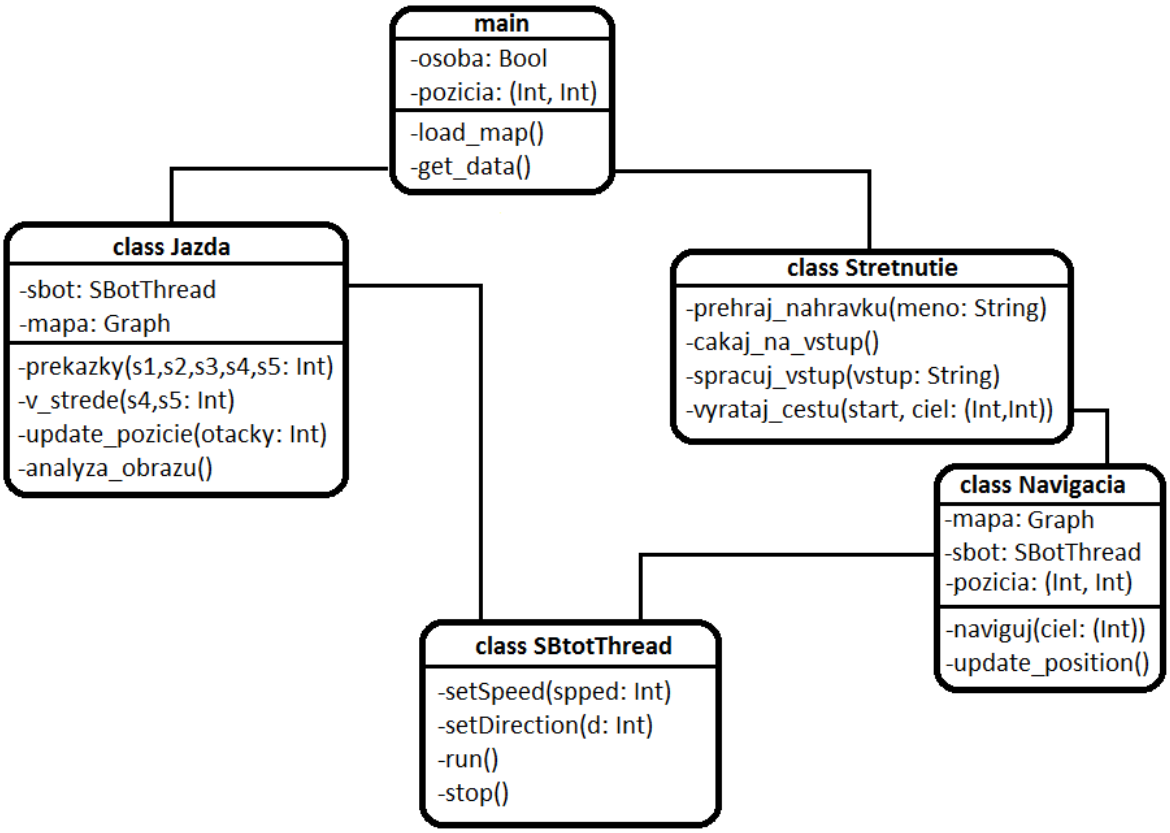
Po úspešnom zadaní vstupu od rozpoznanej osoby sa spustí modul navigácia, ktorý bude postupovať podľa vypočítanej najkratšej cesty.

3.2.2.7 Komponent komunikácia s webom

Tento komponent slúži na komunikáciu s web stránkou na ktorú po ukončení navigácie odošle informácie o navigovaní. Po odoslaní sa znova spúšťa program na jazdenie po pavilóne.

4 Návrh

4.1 Triedny diagram



Obrázok 2 : class-diagram

4.2 Popis tried

4.2.1 Main:

Už naprogramovný modul.

- **load_map()** - nahrá svg súbor s mapou do dátovej štruktúry
- **get_data()** - pomocou naprogramovaných tried get_data dostane z robota údaje o senzoroch a otáčkomere, ktoré ďalej pošle funkciám triedy Jazda

4.2.2 Class SbotThread:

Toto je už naprogramovaná trieda na ovládanie robota.

- setSpeed(speed: Int) – nastaví rýchlosť pohybu
- setDirection(d: Int) – nastaví smer jazdy
- run() - uvedie robota do pohybu

- **stop()** - zastaví robota

4.2.3 Class Jazda:

Kontrola výskytu prekážok, držania smeru jazdy a aktualizácia pozície. Na ovládanie robota využíva už naprogramovanú triedu SbotThread.

- **prekazky(s1,s2,s3,s4,s5: Int)** – v argumentoch dostane čísla zo senzorov, kontroluje výskyt prekážok pred robotom. Ak je prekážka skontroluje podľa bočných senzorov a podľa mapy, či je v rohu.
- **v_strede(s4,s5: Int)** – v argumentoch dostane výstup bočných senzorov, koriguje smer jazdy, tak aby robot šiel čo najviac rovno
- **update_pozicie(otacky: Int)** – pomocou čísla z otáčkomera aktualizuje pozíciu
- **analyza_obrazu()** - analyzuje výstup kamery a pomocou OpenCV zisťuje prítomnosť človeka

4.2.4 Class Stretnutie:

Stretnutie s človekom.

- **prehraj_nahravku(meno: String)** – spustí nahrávku, meno nahrávky dostane v argumente
- **cakaj_na_vstup()** - 60 sekúnd čaká na vstup, potom pokračuje v jazde
- **spracuj_vstup(vstup: String)** – skontroluje korektnosť vstupu, ak je vstup korektný pošle cieľ funkcii vyrátaj_cestu, inak požiada o nové zadanie vstupu
- **vyrataj_cestu(ciel)** – vyráta cestu k požadovanému cieľu

4.2.5 Class Navigácia:

- **naviguj(trasa)** – podľa trasy z argumentu riadi robota do cieľa
- **update_position()** – podľa výstupu otáčkomera aktualizuje pozíciu

5 Testovacie scenáre

Testovacie scenáre sme si rozdelili do dvoch kategórií, všeobecný testovací scenár a testovacie scenáre pre jednotlivé komponenty:

5.1 Testovacie scenáre komponentov:

5.1.1 Reakčný komponent

Po manuálnom spustení robota tlačidlom a po štarte programu by sa mal robot začať pohybovať po pavilóne a snažiť sa rozpoznať postavu. Jeho úlohou je jazdiť dokola a snažiť sa držať v strede chodby.

Testovacie prípady:

- **Hardware a batéria je OK:** Nutnou podmienkou aby robot jazdil po pavilóne je nabitá batéria a funkčné ultrazvukové senzory. V opačnom prípade robot nebude schopný korektnej jazdy.
- **Vstup – rýchlosť (speed, int) :** Pri spustení robota je jedným zo vstupných parametrov rýchlosť robota. Rýchlosť je číslo typu integer v rozsahu <1,10> . V prípade zadania iného čísla alebo nesprávneho typu program vypíše chybu.
- **Vstup – smer (direction, int) :** Ďalším vstupom je smer, ktorým sa robot pohybuje. Smer je defaultne nastavený na priamy pohyb rovno, avšak keď sa robot dostane na koniec uličky musí podľa mapy nájsť správny smer, v ktorom bude pokračovať. V opačnom prípade robot narazí na stenu a zastane.
- **Výstup – senzory (array):** Robot má 5 senzorov, ktoré neustále merajú vzdialenosť od prekážok. Tieto čísla dostane ako výstup typu pole (array). Pomocou bočných senzorov sa robot snaží udržať v strede. Pokiaľ sa hodnota pravého alebo ľavého senzoru konštantne zmenšuje znamená to že robot ide do jednej strany preto sa bude snažiť vyrovnať do opačnej strany. V prípade, že hodnota, niektorého zo senzorov bude menšia ako stanovený limit, znamená to že robot je blízko prekážky. V tomto prípade robot zastane a čaká kým prekážka nezmizne.

5.1.2 Jazdenie po pavilóne

Z reakčného komponentu na vstupe dostáva rýchlosť a smer podľa pozície kde sa robot nachádza. Ak bude smer a rýchlosť správne nastavená nemala by nastať kolízia robota s okolím alebo s prekážkou.

Testovacie prípady:

- Vstup: dobre zvolená rýchlosť a smer Výstup: jazdenie v strede pavilónu bezproblémová detekcia prekážok
- Vstup: príliš veľká rýchlosť , vhodný smer Výstup: možné kolízie, nepresné detekcie ultrazvukových senzorov
- Vstup: vhodná rýchlosť, nesprávny smer : takmer istá nefunkčnosť robota, vysoké riziko kolízie
- Vstup: nevhodná rýchlosť, nevhodný smer Výstup: riziko ujmy na zdravý, možné poškodenie robota či okolitého prostredia

5.1.3 Rozpoznávací komponent

Tento komponent využíva predovšetkým knižnicu OpenCV, pomocou ktorej sa snaží rozpoznať objekt nachádzajúci sa pred robotom. Pomocou ultrazvukových senzorov robot rozporná čí sa pred ním nachádza prekážka. Pokiaľ áno, spustí sa kamera pomocou ktorej zistí či pred ním stojí osoba alebo nie. Po zistení, že pred robotom sa nachádza osoba, komponent odošle informáciu a spustí sa komunikačný komponent .

Testovacie prípady:

Predný ultrazvukový senzor:

- **Cesta voľná** – kamera sa nezapne a robot pokračuje ďalej v jazde a spracovávaní výstupov zo senzoru.
- **Prekážka** – Podľa vzdialenosti od prekážky robot najprv zistí či je potrebné zastať alebo je ešte v dostatočnej vzdialenosti aby nenabúral do prekážky. V momente ako zaznamená prekážku sa zapne kamera aby spracovala obraz pred sebou. Pomocou knižnice OpenCV sa spracuje výstup z kamery a vráti informáciu, či pred robotom stojí osoba alebo čokoľvek iné. Pokiaľ je to osoba tak rozpoznávací komponent pošle informáciu a spustí sa komunikačný komponent. Pokiaľ to nie je osoba robot čaká na odstránenie prekážky.

Predno-bočné ultrazvukové senzory:

- **Cesta voľná** - kamera sa nezapne a robot pokračuje ďalej v jazde a spracovávaní výstupov zo senzorov.
- **Prekážka** - Podľa vzdialenosti od prekážky robot najprv zistí či je potrebné zastať alebo je ešte v dostatočnej vzdialenosti aby nenabúral do prekážky. V momente ako zaznamená prekážku sa zapne kamera aby spracovala obraz pred sebou. Pomocou knižnice OpenCV sa spracuje výstup z kamery a vráti informáciu, či pred robotom stojí osoba alebo čokoľvek iné. Vzhľadom na to, že objekt v tomto prípade nemusí byt celý

nasnímaný kamerou a teda, že knižnica OpenCV ho nedokáže spracovať. V takomto prípade sa vyhodnotí objekt pred robotom za prekážku a teda robot bude čakať na odstránenie. Pokiaľ OpenCV rozpozná postavu tak rozpoznávací komponent pošle informáciu a spustí sa komunikačný komponent.

Bočné ultrazvukové senzory:

- **Cesta voľná** – v tomto prípade nastane len pokiaľ sa robot nachádza na križovatke chodieb. Kamera sa nezapne a robot pokračuje v jazde smerom, ktorým sa sám rozhodne na základe mapy.
- **Prekážka** – v tomto prípade sa kamera zapínať nebude. Pretože bočné senzory slúžia na udržiavanie robota v strede chodby. Pokiaľ nastane situácia, že osoba prejde popri robotovi zprava alebo zľava tak robot v tomto prípade nereaguje a považuje človeka za stenu.

5.1.4 Komunikácia so serverom

Tento komponent slúži na komunikáciu s web stránkou na ktorú po ukončení navigácie odošle informácie o navigovaní. Po odoslaní sa znova spúšťa program na jazdenie po pavilóne.

Testovacie prípady:

- **Vstup : send_stav(stav: string)**: Počas celej navigácie robot posiela na server informácie o svojej pozícii, ktoré sa následne zobrazujú robota na mapke. V prípade zadania nesprávneho stringu alebo iného typu server nebude schopný spracovať údaje a vypíše chybové hlásenie, poprípade vykreslí neaktuálnu mapu.
- **Vstup : send_quest(quest: string)**: Po skončení navigácie sa na server odošlú informácie o úlohe. V prípade odoslania nesprávneho typu informácií server vypíše chybové hlásenie a nebude možné skontrolovať splnené úlohy robota.
- **Vstup: recv()** : Cez server je možné aj zapnutie a vypnutie robota. V prípade, že by nastali nejaké problémy so serverom nebude nijako ohrozená samotná funkčnosť robota. Pri odoslaní nekorektných dát na zapnutie či vypnutie robota sa robot bude ovládať priamo na mieste vykonávania činnosti.

5.1.5 Navigačný komponent

Po stretnutí s človekom a po zadaní korektného vstupu, by robot mal vyhľadať najkratšiu cestu k požadovanému cieľu. Následne by sa mal začať pohybovať podľa vypočítanej trasy až kým nedorazí do cieľa.

Testovacie prípady:

Korektný vstup – aby sa dala nájsť najkratšia cesta, je nutné aby robot dostal správny vstup, teda číslo existujúcej miestnosti v správnom formáte.

- najdi_cestu(ciel: Int)
  - **vstup** – číslo zodpovedajúce niektorej miestnosti v pavilóne
  - **výstup** – najkratšia cesta od aktuálnej polohy k zadanej miestnosti vo formáte poľa čísiel vrcholov (pravdepodobne)
  - **vstup** – hocijaký nečíselný vstup, alebo číslo, ktoré nezodpovedá žiadnej miestnosti v pavilóne
  - **výstup** – program spadne na chybe, avšak zlý vstup by sa sem dostať nemal
- **update_position()**
  - **vstup** – hodnota nameraná otáčkomerom: podľa hodnoty, ktorú ako výstup dá otáčkomer sa aktualizuje pozícia. Otáčkomer by mal dávať korektný výstup, v opačnom prípade program spadne na chybe
- **naviguj (cieľ (int,int))**
  - **vstup** – trasa k cieľovej miestnosti ako pole ID-čiek miestností
  - **výstu**p – funkcia robota ovláda a naviguje podľa vstupne trasy
    nekorektný vstup – hocičo iné ako pole ID-čiek miestností, alebo pole obsahujúce nekorektné ID, potom na takomto vstupe program spadne na chybe