ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra kybernetiky

Predikce akcí uživatele podle odpozorovaného chování

Bakalářská práce

27. 5. 2011

Autor: Václav Černý

Vedoucí bakalářské práce: Ing Petr Novák

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 27.5.2011

Václav Čirný

Podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Václav Černý

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Predikce akcí uživatele podle odpozorovaného chování

Pokyny pro vypracování:

1. Seznamte se s existujícími návrhy / realizacemi pro predikci chování uživatele ve vytvořených domácích inteligentních řídicích systémech (obsluhy domácího prostředí). Případně s dalšími teoretickými návrhy / pracemi.
2. Zhodnoťte jejich výhody a nevýhody zejména z pohledu uživatele s omezenou schopností pohybu a reakce. Rovněž se zaměřte na poskytovanou účelnost, adaptivitu a případně další charakteristické parametry.
3. Seznamte se s algoritmy / postupy, které mohou být využity jako predikční části pro inteligentní řídicí systémy (nejen v) domácnosti, a zhodnoťte jejich užitnost / vhodnost / náročnost.
4. Vyberte vhodný algoritmus a pomocí něj navrhněte řešení dané problematiky. Zaměřte se zejména na maximalizaci úspěšnosti predikce a detekci posloupností / opakujících se akcí.
5. Vytvořte modul / knihovnu obsahující navržené řešení, jejímž vstupem budou akce průběžně vykonávané uživatelem a současně některé stav okolního prostředí (základní veličiny jako světlo, teplo, čas) a výstupem bude seznam predikovaných akcí seřazených podle určitých kritérií (například pravděpodobnost, návaznost na jiné).
6. Činnost vytvořeného modulu / knihovny ověřte minimálně na souboru „uměle" vytvořeného logu akcí (fiktivního uživatele). Zhodnoťte výstup pomocí některých parametrů, jako jsou úspěšnost predikce nebo rozpoznání posloupností vykonávaných akcí.

Seznam odborné literatury:

[1] Internetové stránky prodejců / výrobců domácích řídicích systémů.
[2] Další dostupné informace o této problematice (internet publikace / diplomové a bakalářské práce).
[3] Knihy / dokumentace Microsoft .NET Framework, jazyk C# 2005, 2008, 2010) a jeho komponenty / části.
Další literaturu / podklady dodá vedoucí práce.

Vedoucí bakalářské práce: Ing. Petr Novák

Platnost zadání: do konce zimního semestru 2011/2012

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 4. 2. 2011

Abstrakt

Bakalářská práce, kterou držíte v rukou, se zabývá využitím softwarového nástroje pro predikci akcí uživatele v rámci inteligentního prostředí a inteligentních domácností. Tento nástroj by měl sloužit především pro zlepšení života uživatelů s omezenou schopností pochybu a reakce. Predikce by měla usnadnit a zrychlit uživateli pohyb, pobyt a vykonávání činností v prostředí inteligentního domova. Jedním z cílů této práce bylo také navrhnout softwarovou knihovnu/modul, která na základě záznamu činností z minulosti predikuje akce, které nejlépe vyhovují současnému stavu, jako je poloha uživatele, čas a posloupnost předchozích akcí. Celá knihovna včetně všech metod byla implementována v programovacím jazyku C# z platformy .NET . Pro implementaci bylo využito integrované vývojové prostředí Microsoft Visual Studio 2010.

Abstract

Bachelor thesis, which you are holding in front of you, deals with the usage of software tool for user's actions prediction in smart home and smart environments. This tool should serve to improve life of users with limited mobility and reactions. Prediction should simplify and make faster user's movement, stay and performance in the smart home environment. One of goals of this thesis was also to design an software library/modul which, on the basis of record of activity done it the past, predicts actions which suit best to current state like user's position, time and sequence of previous actions. Whole library including all of the methods was implemented in programming language C# from platform .NET . Microsoft Visual Studio 2010 Integrated development environment was used for the implementation.

Obsah

1. Úvod

Pro osoby se sníženou schopností pohybu a schopností reakce je mnohdy problém vykonávat úkony snadné pro běžného člověka a spojené s jeho standardními potřebami, jako je pohyb po svém vlastním domově / bytě a interakce s obvyklými elektrickými přístroji. Se současnou mírou pokroku a rozvoje výpočetní techniky, umělé inteligence a komunikačních technologií je vhodné pokusit se i těmto lidem pomoci tak, aby mohli žít plnohodnotný a soběstačný život a usnadnit jim ovládání mnoha běžných zařízení. Tito lidé to navíc často potřebují mnohem více než většina populace. Snahou této bakalářské práce je navrhnout knihovnu, která by tuto „pomoc“ simulovala a to v podobě předpovědi akcí. Výsledný modul bude součástí většího uceleného systému vyvíjeného v rámci skupiny NIT (Nature Inspired Technology) na katedře kybernetiky, zabývající se aplikacemi pro možnost řízení domácího prostředí.

2. Inteligentní domy

2.1. Automatické systémy

Oblast automatizace systémů je, jak už přímo z názvu vyplývá, poměrně široké odvětví. Slovo systém je pojem velmi rozsáhlý a nevymezuje zcela přesně, co ještě pod něj lze nebo nelze zahrnout. Podíváme-li se například do otevřené encyklopedie Wikipedia [8] pod klíčové slovo „Systém“, spatříme následující vymezení: „Systém (česky soustava) je souhrn souvisejících prvků, sdružený do nějakého smysluplného celku. V latině a řečtině znamená termín system *kombinovat, uspořádat, sdružovat*. Systém se obvykle skládá z *komponent* (nebo *elementů*), které jsou spojeny za účelem umožnění toku informací, materiálu nebo energie. Termín je často používán pro popis entit, které se vzájemně ovlivňují a pro něž může být vytvořen matematický model. *Subsystém* je systém, který je částí jiného systému.“

Slovo automatický znamená českým slovem samočinný, též mechanický. Zapátráme-li po termínu automatizaci, nalezneme například, že automatizace „označuje použití řídicích systémů (např. regulátorů, počítačů, snímačů) k řízení průmyslových zařízení a procesů. Z pohledu industrializace jde o krok následující po mechanizaci. Zatímco mechanizace poskytuje lidem k práci zařízení, které jim usnadňuje práci, automatizace snižuje potřebu přítomnosti člověka při vykonávání určité činnosti. Za splnění ideálního předpokladu tzv. komplexní automatizace by teoreticky mohlo dojít až vyřazení člověka z příslušného výrobního procesu. V praxi se prozatím jeví tato možnost jako neuskutečnitelná.“

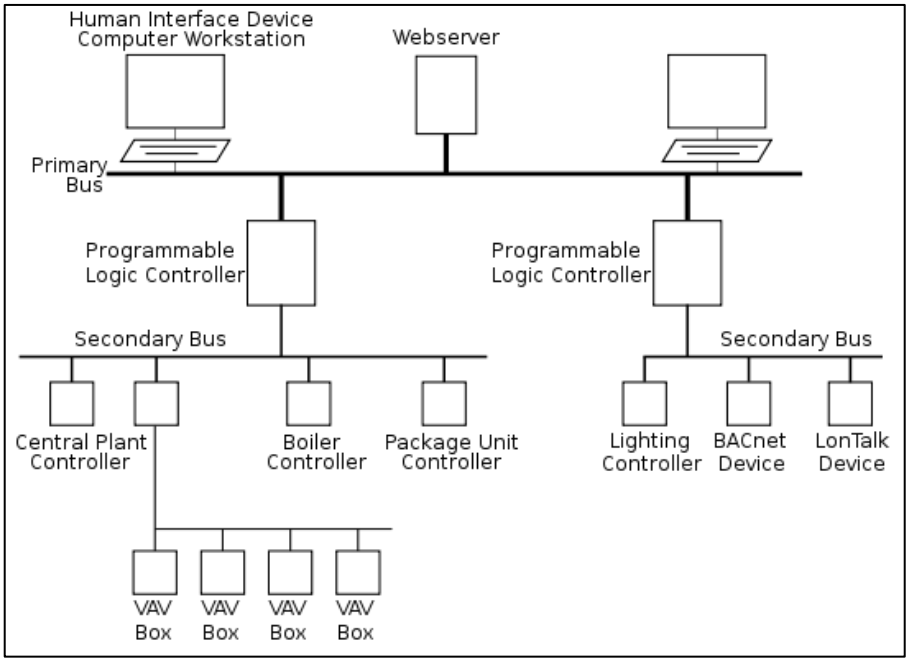
Důležitá je část říkající: „… automatizace snižuje potřebu přítomnosti člověka při vykonávání určité činnosti. …“. Otázkou však zůstává, jak tohoto co nejlépe dosáhnout.

Pomineme-li teď paranoidní nebo katastrofické scénáře vidící vyřazení homo sapiens sapiens z výrobního cyklu jako krok k uvrhnutí lidské rasy do otroctví strojů. Na vyloučení člověka z procesu by se z určitého úhlu pohledu dalo při splnění podmínek komplexní automatizace nahlížet jako na neskonalé pohodlí, protože by si člověk ušetřil velkou práci. Nakonec i to je jedním z důvodů, proč se člověk snaží vyvíjet stále lepší a lepší stroje nebo sofistikovanější systémy.

2.2. Automatizace domů

Dobrým příkladem automatizace systémů v konkrétních případech je právě automatizace budov, případně automatizace domácností. Existuje celá řada společností, které se automatizací například rodinných domů zabývá. Ve většině případů jde ovšem o automatizaci, která se neobejde bez lidského faktoru, byť využívá ve značné míře nejmodernější techniky. Jedná se zejména o kontroléry a řídící jednotky zajišťující spínání topení, nastavení pokojové teploty, dále jednotky řídící úroveň osvětlení, spouštění žaluzií, zakrývání bazénu nebo bezpečnostní systémy chránící objekt. Ovládání se vykonává pomocí dotykových nebo tlačítkových panelů umístěných na zdi na dobře dostupném místě. V lepším případě je vše integrováno do jednoho panelu, z kterého se může uživatel proklikat přes menu až na požadovanou technologii a nastavit ji podle aktuální potřeby. Stále k ní však musí fyzicky přistoupit, vzít do ruky ovladač nebo poslat sms zprávu z mobilního telefonu a technologii nastavit. U některých komponent, jako je úroveň osvětlení nebo topení, lze dodatečně využít dalších externích senzorů měřících venkovní stav světla nebo venkovní teplotu. Na základě údajů z těchto senzorů se pak samy dokážou ve správný okamžik sepnout. Uživatel tak může například ráno vstávat s postupným rozsvěcením světel nebo nechat vytopit byt na příjemnou teplotu zatímco se vrací z práce nebo dovolené.

Mezi takto propojené jednotky patří topení nebo klimatizace, osvětlení, žaluzie, alarm, garážová vrata a podobné. Jedná se prakticky o distribuované systémy,



Obr. 1 Schéma řízení domácnosti [8]

kdy centrální panel pouze rozesílá konfiguraci nastavení, nanejvýše si toto nastavení pamatuje. Samotné řízení elektrického zařízení provádějí u každé technologie řídící jednotky.

Při prohledávání současných nabídek ([1], [3]) společností zabývajících se tímto odvětvím jsem ale vždy narazil na více či méně podobný a omezený sortiment standardních funkcí:

- ovládání světel a intenzity osvětlení
- spínání světel po vstupu do místnosti
- ovládání regulace topení, spínání boileru v závislosti na čase, venkovní a pokojové teplotě
- bezpečnostní systémy
- možnost ovládání pomocí sms z mobilu nebo pomocí chytrých telefonů
- energetická optimalizace, šetření energií

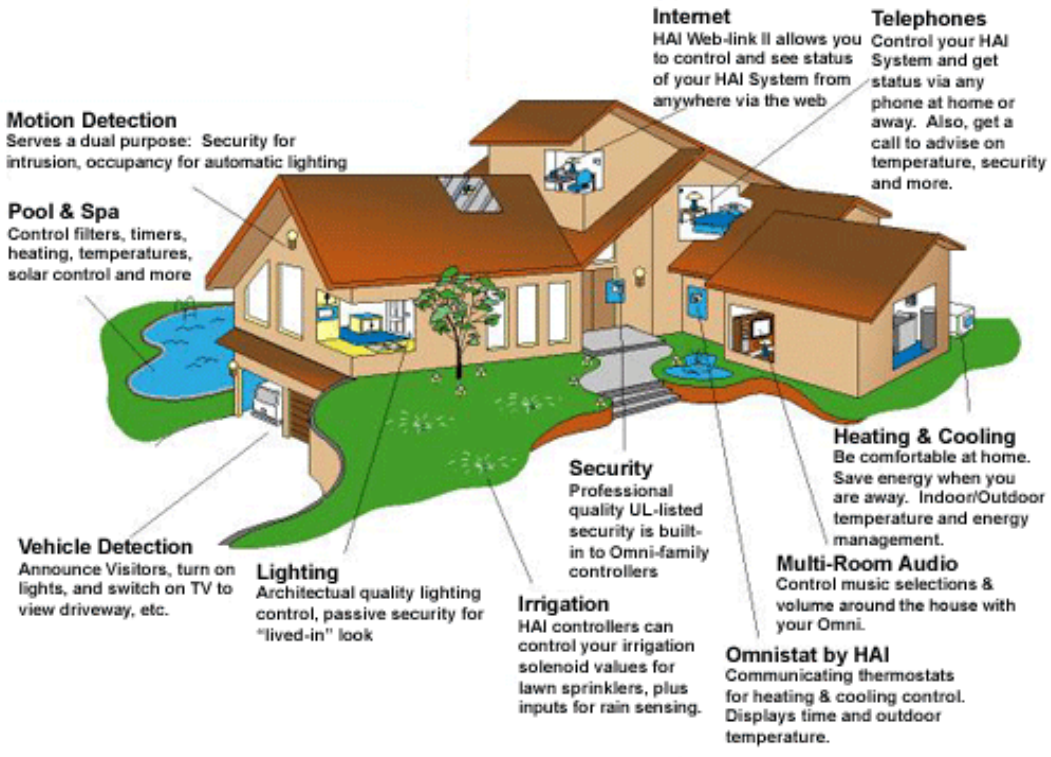
Do tohoto systému většinou nespadají zařízení umístěná například v kuchyni, jako jsou mikrovlnná trouba, lednice nebo sporák. Všechna elektrická zařízení jsou obvykle propojena standardně pomocí kabelů. V případě, že by uživatel chtěl takto upravit svůj příbytek po dostavění domu, nebo zabydlení bytu, je nutné uskutečnit zásahy do prostorů interiéru, protože kabely je potřeba někam umístit, pokud možno alespoň tak, aby nepůsobily rušivým dojmem.

2.3. Smart Home

Krokem kupředu jsou takzvané smart home domy, nebo smart home environment neboli inteligentní domy. Inteligentní dům se na první pohled zvenku neliší od normálního domu. Uvnitř je ale budova vybavená speciální strukturou počítačové, komunikační techniky, celou řadou nejrůznějších typů senzorů a elektrických zařízení schopných ovládat a programovat inteligentní domácí systémy podle potřeb obyvatel a tím zvyšovat jejich pohodlí.

Koncept chytrých domů pochází ze Spojených států, odkud se rozšířil do západní Evropy a získal zde i podporu Evropské unie. EU také přímo v Bruselu postavila jeden prezentační dům (www.livingtomorrow.be). U nás v České republice se nacházejí dva ukázkové inteligentní rodinné domy a jeden zahradní domek. Vše v Praze 4.

Myšlenka chytrých domů je na světě již pár let. Jeden z prvních inteligentních domů nechal postavit i sám zakladatel společnosti Microsoft Bill Gates. Šlo především o demonstraci, co vše je možné a kudy by se blízká budoucnost v oblasti budov mohla ubírat a jak by jednou mohli naše domovy vypadat. Důvodem, proč nebyl Gates okamžitě následován, nebyla jen cena ve výši dva milióny amerických dolarů, nýbrž také fakt, že v té době ještě neexistoval dostatek produktů a technologií pro takovýto typ domů.



Obr. 2 Ilustrace funkčních součástí inteligentního domova

Další problém, který nebylo možné přehlédnout, byla nutnost propojení pomocí kabelů a drátů všech subjektů zamýšlených do systému inteligentního domu. Při počtu technologií, které člověk v domácnosti dnes využívá, by objem kabeláže nebyl nijak

zanedbatelný. Spolu s tím i skutečnost, že je potřeba kabely někam umístit. Vedle toho rovněž další skutečnost, že v té době mikrovlnná trouba, sporák nebo alarm nedokázaly komunikovat s počítačem nebo jinou nadřazenou sofistikovanější jednotkou.

V dnešní době už je situace poněkud odlišná. Spolu s poklesem ceny výpočetní techniky a tedy i počítačů došlo k rozvoji standardu 802.11 neboli technologie Wi-fi. Společnost IBM označila tuto technologii za „zachránce“ programu chytrých domů a rozhodla se pro rozsáhlou investici ve výši deset miliard amerických dolarů do programu chytrých domů. I tento akt napovídá, kam bude budoucí vývoj budov směřovat. Implementace levného a malého wi-fi čipu do elektrického spotřebiče z něj dokáže udělat přístroj, který může vysílat informace o svém stavu nebo přijímat hodnoty nastavení. Wi-fi tedy zajistí obousměrnou komunikaci a vyřeší problém umístění kabelů. Výhodou technologie wi-fi je také to, že se ke standardu 802.11 přiklonily všechny přední firmy z oblastí komunikací a informačních technologií jako jsou Intel, Microsoft, Cisco, Samsung nebo Sony.

Své uplatnění v chytrých domech najdou jistě i interaktivní displeje integrované například do zrcadel, skleněných výplní dveří, stěn pokojů anebo zabudované do kuchyňské linky, tak jak to ukázala společnost Microsoft ve svém projektu Future Home. Vývoj v oblasti mobilních telefonů a počítačů jde také směrem ke kapacitním dotykovým displejům, tak proč nemít zabudovaných několik počítačů se speciálními funkcemi i v různých částech bytu a pomocí naučených „chytrých“ gest s nimi ovládat přístroje v jiných místnostech. Ovládací technologie Kinect vyvinutá primárně pro herní průmysl na konzoli Microsoft Xbox 360 by své uplatnění jistě také našla. Jedná se o technologii, kdy je hráč sledován kamerou umístěnou nad televizní obrazovkou, jeho pohyby jsou rozeznávány pomocí softwaru a přenášeny „naživo“ do herního prostředí, aniž by musel v ruce držet ovladač, klávesnici nebo myš. Kinect je testován například na fakultě Biomedicínského inženýrství jako prostřed pomáhající při rehabilitaci pacientů.

2.4. Predikce činností uživatele

Dalším logickým stupněm a zejména způsobem jak ještě více zlepšit a zdokonalit chytré domy, je snížení jejich přímé závislosti na ovládání uživatelem na co nejnižší možnou míru. Tedy udělat pobyt ještě více pohodlnější a vytvořit z něj opravdu místo

relaxace a odpočinku. To znamená s největší možnou mírou pravděpodobnosti dokázat odhadnout a předpověď, co by obyvatel nebo obyvatelé mohli v daný časový okamžik udělat a eventuálně tento úkon přímo vykonat místo člověka. Případně mu tuto činnost alespoň navrhnout na nejbližší panel v jeho dosahu, aby mohl jen snadno potvrdit její vykonání. Ideálním stavem by samozřejmě bylo umět předpovědět každou činnost uživatele a vykonat ji místo něj bez toho, aby se systém dotazoval. To může fungovat bezchybně pouze v případech, kdy obyvatel dělá pravidelně činnosti den po dni a své zvyky prakticky nemění, včetně času jejich vykonání. Systém nedokáže předpověď, kdy bude chtít obyvatel domu změnit svůj zvyk a znenadání dělat jinou akci než dělal doposud. Inteligentní dům je stále jen stroj, který se dokáže „učit“ pouze do omezené míry.

Tím se dostáváme k základnímu problému tzv. inteligentního domu a jeho případnému rozšíření o schopnost predikce činností uživatele. A to k tomu, že inteligentní dům není inteligentní v celém významu slova inteligence. Vyberme některou z možných definic pojmu inteligence:

"Inteligence je všeobecná schopnost individua vědomě orientovat vlastní myšlení na nové požadavky, je to všeobecná duchovní schopnost přizpůsobit se novým životním úkolům a podmínkám."*(Němec William Stern)*

Přidejme vymezení inteligence z jiného zdroje, tak jak ho uvádí wikipedie :

„Inteligence (z lat. *inter-legere*, rozlišovat, poznávat, chápat) je rozumová schopnost řešit nově vzniklé nebo obtížné situace; schopnost učit se ze zkušeností; schopnost přizpůsobit se; ….. “. A dále pokračuje: „Je to vlastnost, což znamená, že je vrozená, nemůžeme tedy její míru ovlivnit, ale můžeme ji rozvíjet získáváním zkušeností a procvičováním modelových situací.“

Smart House splňuje pouze část této definice a to schopnost „učit ze zkušeností“ a „řešit“ vzniklé situace. Zaměříme-li se na druhou část zmiňující „…schopnost přizpůsobit se novým … úkolům a podmínkám…“ a „…schopnost řešit nově vzniklé nebo obtížné situace…“ zjistíme zaostávání inteligentního domu ve svých schopnostech. Není v jeho silách předpovědět akci s pravděpodobností rovnající se jistotě. Stále má ovšem schopnost do nemalé míry usnadnit člověku pobyt mezi jeho

zdmi. Můžeme od něj očekávat jakýsi návrh akce nebo seznamu více akcí, které bychom mohli chtít v daný časový okamžik vykonat.

Je ale dobré znát jeho omezení a slepě ho nenechat rozhodovat o každé činnosti, kterou bychom eventuálně mohli vykonat. Absolutní rozhodovací pravomoc systému nanejvýše svěřit u „bezpečných“ a především jistých akcí, tak jak o tom rozhodují současné chytré domy, tedy regulace topení, osvětlení a bezpečnostní zajištění objektu.

2.5. Využití inteligentních domů pro osoby s omezenou schopností pohybu a reakce

V současné době není na trhu mnoho systémů, které by osobám s omezenou schopností pohybu a reakce nebo podobně postižených usnadňovaly jejich každodenní život a pobyt v jejich domovech. Existuje sice poměrně značné množství pomůcek usnadňující hendikepovaným lidem například pohyb po ulici a venkovních prostorách nebo usnadňují dopravu aj. Čeho je ale nedostatek, jsou systémy zabývající se zvýšením kvality života uvnitř domů nebo bytů. Pro uživatele s omezenou pohyblivostí je často problém zapnout si standardní domácí spotřebiče ať už v kuchyni, jakou jsou mikrovlnná trouba, sporák či rychlovarná konvice, nebo v obývacím pokoji, kde by si třeba rádi otevřeli okno, zapnuli klimatizaci anebo večer zatáhli závěsy. K většině těchto úkonů potřebují pomoc buď přímo přiděleného ošetřovatele, nebo rodinného příslušníka, případně známého.

Chytré domy s rozšířením na predikci akcí by měly velké uplatnění právě v životě lidí tělesně postižených nebo jinak pohybově nebo reakčně omezených. Mohlo by se rovněž jednat například jen o omezený, ale delší časový interval, například při nutnosti setrvat v domácím ošetřování po vážné operaci. Cílem této bakalářské práce je navrhnout právě takový systém, který by lidem s omezenou schopností pohybu nebo reakce pomohl a usnadnil pobyt doma, případně v jiném prostoru, kde často a dlouhodobě setrvávají. Většinu elektrických spotřebičů, počítač, televizi, rádio, ovládání světel a další již ve svém obydlí mají, jde jen o to propojit vše vhodně dohromady a zavést mezi těmito technologie komunikační kanály. Tedy spojit vše do jednoho

systému, který by bylo možné centrálně řídit z jednoho bodu – počítače – umístěného například na opěradle invalidního vozíku. Kromě ovládání domácnosti by byl rovněž schopen kontrolovat i pohyb vlastního elektrického vozíku.

Podobný systém je vyvíjen na katedře kybernetiky pod skupinou NIT – Nature Inspired Technologies Group, jejíž součástí je i vedoucí mé bakalářské práce Ing. Petr Novák. Součástí systému by měla být i softwarová knihovna / modul vytvářená v rámci této práce.

3. Cíle

Cílem bakalářské práce je seznámit se s možnostmi predikce akcí uživatele v domácích inteligentních řídicích systémech neboli inteligentních domech a prozkoumat, jaká je současná situace v tomto oboru. Případně jaké systémy jsou na trhu nabízeny a zjistit jejich výhody a efektivitu, případně nevýhody a nedostatky zejména pro uživatele s omezenou schopností pohybu a reakce.

Dále se seznámit s postupy nebo algoritmy, které by mohly být využity pro predikční modul v inteligentních řídicích systémech například v domácnostech, a zvolit postup vhodný nejen pro predikci jednotlivých akcí ale i jejich posloupností tak, aby bylo dosaženo pokud možno co největší vhodnosti předpovědi. Cílovou skupinou uživatelů pro využití vytvářené knihovny je zejména skupina osob s omezenou schopností pohybu a reakce. Tedy například lidé na invalidním vozíku, jinak tělesně postižení, nebo starší lidé, potřebující často pomoc ošetřovatele nebo rodiny.

Hlavní snahou této práce je vytvořit modul neboli softwarovou knihovnu v programovacím jazyku C#, která bude zvolenou činnost vykonávat. Vstupem knihovny bude vždy akce, kterou uživatel právě vykonal. Výstupem naopak návrh několika akcí, případně posloupnost akcí, které na základě navrženého algoritmu připadají nejvíce v úvahu jako akce, které by mohl uživatel chtít následně vykonat. Výstupní akce budou seřazeny podle určitých kritérií, především podle jejich pravděpodobnosti (předpokladu použitelnosti), jak mohou být použity. U sekvencí podle návaznosti na předcházející (již vykonané) akce. Akcí se rozumí interakce uživatele s některým zařízením, které lze ovládat dálkově pomocí elektroniky, nebo počítače. Od původního záměru zahrnout do systému predikce i stavy okolního prostředí (jako jsou venkovní teplota, nebo úroveň osvětlení) bylo nakonec ustoupeno, neboť tato skutečnost nebyla shledána určující pro predikci. Jinými slovy existuje jen minimum akcí, které by byly těmito vlivy přímo ovlivněny. Stavem, na který byl naopak brán zřetel, je čas.

Celá činnost knihovny bude testována na souboru uměle vytvořených logů akcí imaginárních osob, které ale do značné míry simulují realitu a odpovídají skutečnému chování lidí ve svých bytech, nebo domech. Jako základní log byla vybrána osoba pohybující se po svém bytě na invalidním vozíku a ovládající různé spotřebiče pomocí speciálního komunikátoru (umístěného na vozíku).

4. Návrh

V této kapitole je obecně popsán způsob řešení predikce akcí tak, jak je navržen logický model pro další implementaci softwarové knihovny. Popis je ve stejném pořadí v jakém postupuje i algoritmus knihovny.

4.1. Sbírání dat

Nejprve je potřeba knihovnu naplnit informacemi, ze kterých bude vycházet a na základě kterých se bude rozhodovat. Je tedy potřeba určitou dobu před uvedením prediktivní funkce do provozu sbírat údaje o činnosti obyvatele domu, nebo bytu, který bude knihovnu používat. Do textového souboru zaznamenávat čas, pozici, respektive místnost, a akci, která v ní byla vykonána. Akcí je myšleno použití zařízení, která jsou zapojena do celého systému inteligentní domácnosti a která lze tedy i vzdáleně ovládat. Výše zmíněný textový soubor uchovávající hodnoty činností / akcí uživatele se nazývá log.

Pokud je zaznamenáno dostatečné množství vstupních dat o činnosti obyvatele v jeho prostředí, můžeme začít používat predikci vhodných akcí. Bez množiny dat není reálně možné úspěšně vykonávat predikci a celý proces nebude mít odpovídající výsledek. Je potřeba nejprve výpočetní jednotku naplnit daty.

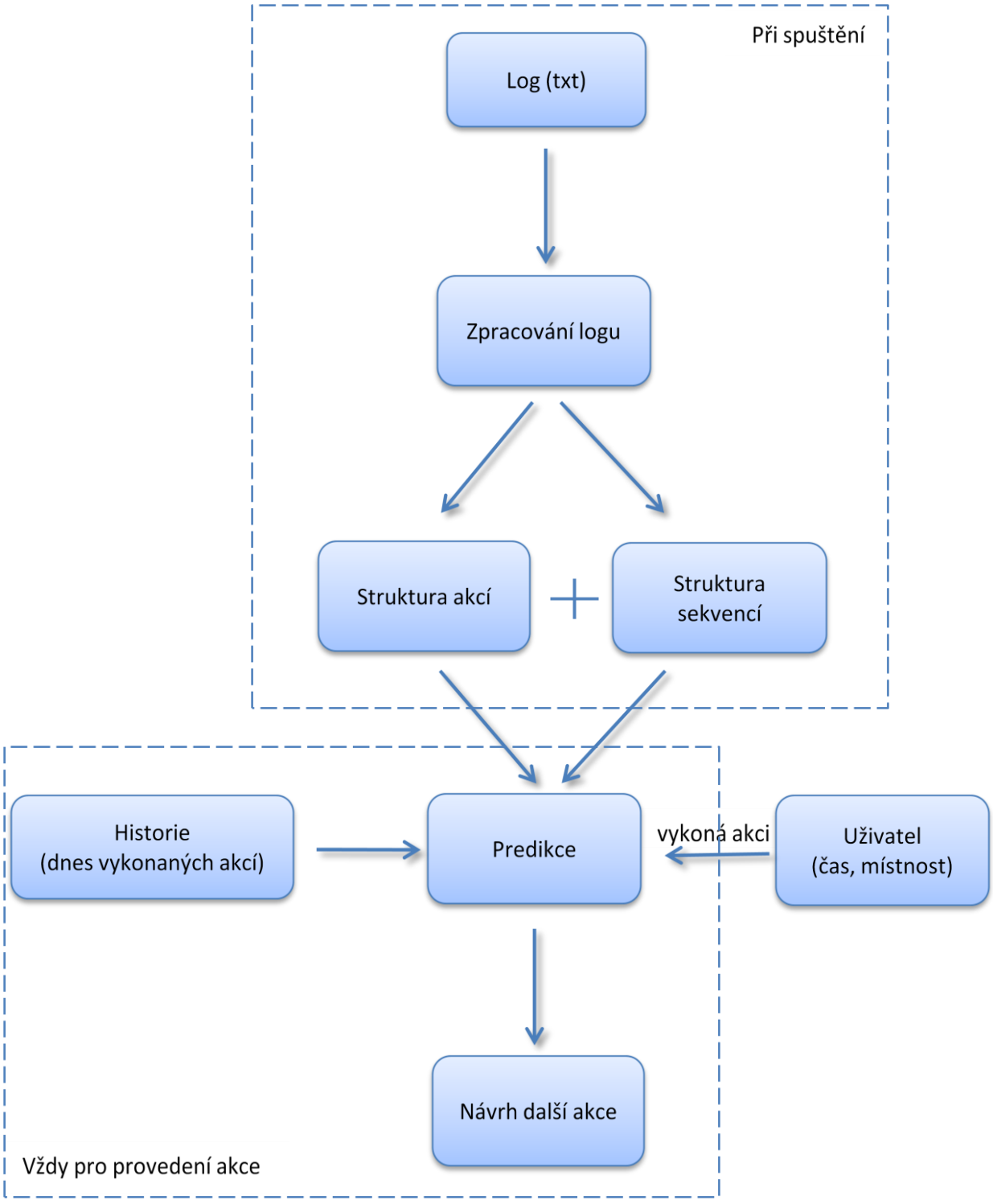
Princip práce softwarové knihovny vytvářené v rámci této bakalářské práce je popsán detailně v následujících kapitolách. Pro prvotní nabytí obecné představy probíhá logika knihovny v následujících krocích (viz obr. 3):

- Získání záznamů z domácnosti o pohybu a činnosti uživatele
- Předzpracování záznamů
    - Průzkum logu a vygenerování časových intervalů
    - Detekce akcí a místností užívaných uživatelem

        - Vygenerování struktury reflektující četnost vykonaných akcí v závislosti na čase a pozici / místnosti
        - Vygenerování struktury uchovávající sekvence akcí, jejich četnost a podobnost s ostatními sekvencemi

- Vlastní predikce
    - Zjištění času
    - Zjištění aktuální pozice / místnosti uživatele
    - Predikce možných akcí a sekvencí

Původním záměrem bylo zaznamenávat do logu i hodnoty některých dalších okolních stavů, jako jsou úroveň venkovního světla a venkovní teplota. V průběhu implementace od toho však bylo upuštěno, protože nebylo nalezeno mnoho akcí, které by přímo souvisely s těmito okolními stavy nebo by jimi byly ovlivňovány. Respektive nebylo dostatečně doloženo, zda je uživatel vykoná nebo ne nevykoná na základě těchto okolních stavů. Demonstrujeme-li toto na příkladu, tak kupříkladu skutečnost, jestli si uživatel zapne rádio nebo televizi nezávisí na úrovni venkovního osvětlení ani na tom, zda je venku 5 nebo 30 °C. Z tohoto důvodu tyto stavy nebyly brány v úvahu, a proto ve výsledném programu nejsou zahrnuty.



Obr. 3 Způsob práce knihovny

4.2. Zpracování naměřených dat

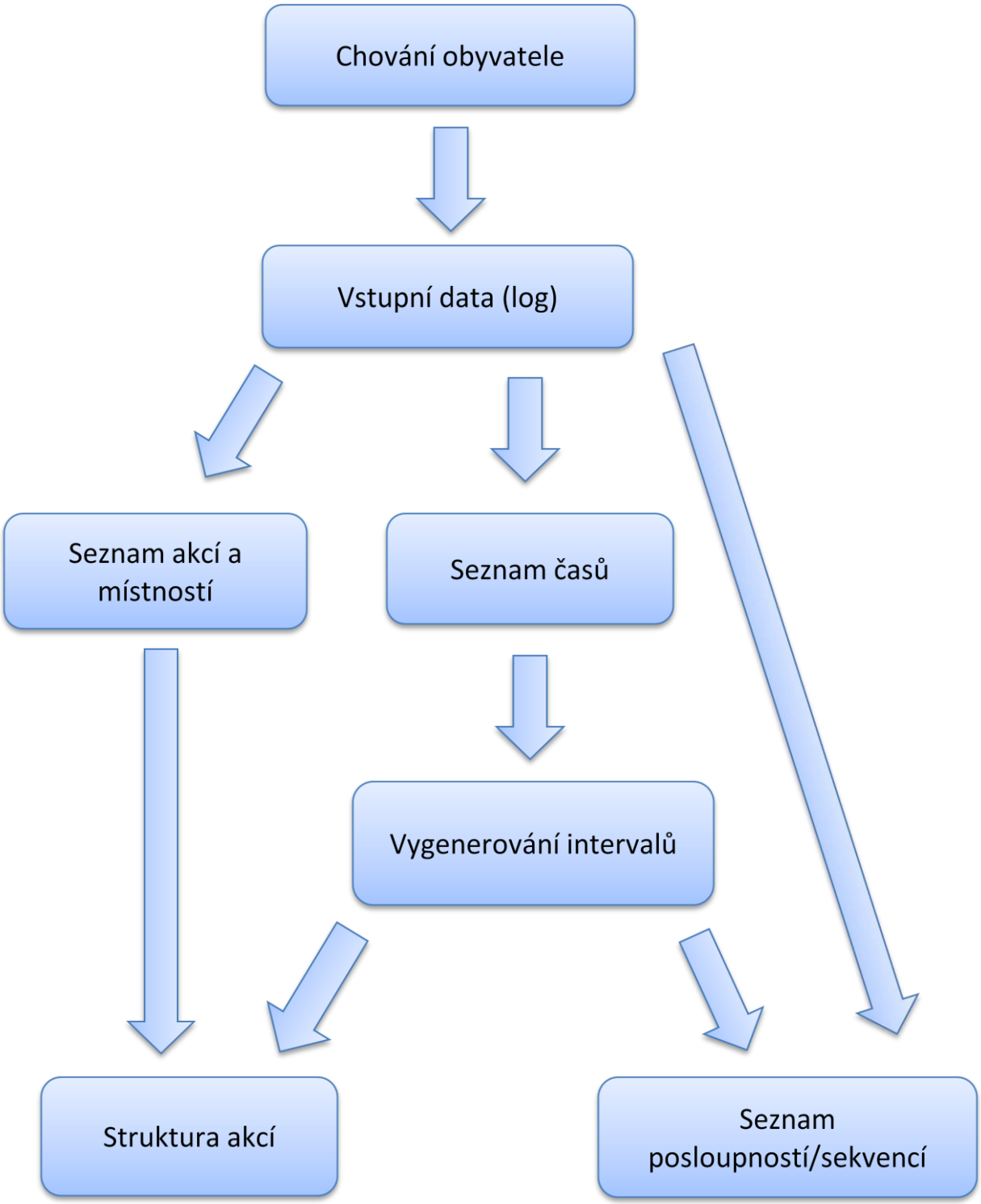
Před samotným používáním knihovny se, buď manuálně, nebo automaticky, spustí z řídícího počítače, zabudovaného například na opěradle elektrického vozíčku, část modulu, která zpracuje data z logu získaná v rámci mapování činnosti uživatele za určité časové období. Vhodným okamžik pro vykonání této části jsou buď brzké ranní hodiny, respektive noc po půlnoci. Eventuelně dostatečně včas před vstáváním uživatele. Výpočet, který modul uskuteční, není nikterak výpočetně náročný, takže doba jeho zahájení není kritická. Záleží samozřejmě na výkonnosti použité výpočetní jednotky. Tato část vykoná nejprve tři důležité úkony (viz obr. 4) :

- Rozdělení dne do vhodných časových intervalů v závislosti na četnosti akcí v jednotlivých částech dne
- Roztřídění akcí v závislosti na místě a době konání a zhodnocení jejich výskytu
- Nalezení sekvencí akcí, jejich četností a podobností s jinými sekvencemi

Nyní konkrétněji k jednotlivým částem. Nejprve se provede průzkum logu z hlediska času, ve kterém došlo k jednotlivým událostem. Byť bychom našli velmi konzervativního uživatele, který by dělal každý den vždy ve stejný čas stejnou akci a takto to dokázal opakovat den po dni, není ve skutečnosti možné, aby uživatel, vykonával akce v přesně stejnou dobu. Je proto praktické rozdělit časovou osu na určité vhodné intervaly a Každé akci pak z logu přiřazovat interval, ve kterém se ve skutečnosti udála. Není ale nutné rozdělit den na určitý počet stejně dlouhých intervalů a takové řešení by nebylo úplně přesné. Je vhodnější zjistit z nahraného logu, v jakých částech dne dochází k větší koncentraci akcí. Jinými slovy řečeno, kdy je uživatel aktivnější a naopak které části dne jsou z hlediska domácí elektroniky využité jen velmi málo nebo dokonce vůbec. A tomuto poznatku pak přizpůsobit rozdělení dne do časových intervalů.

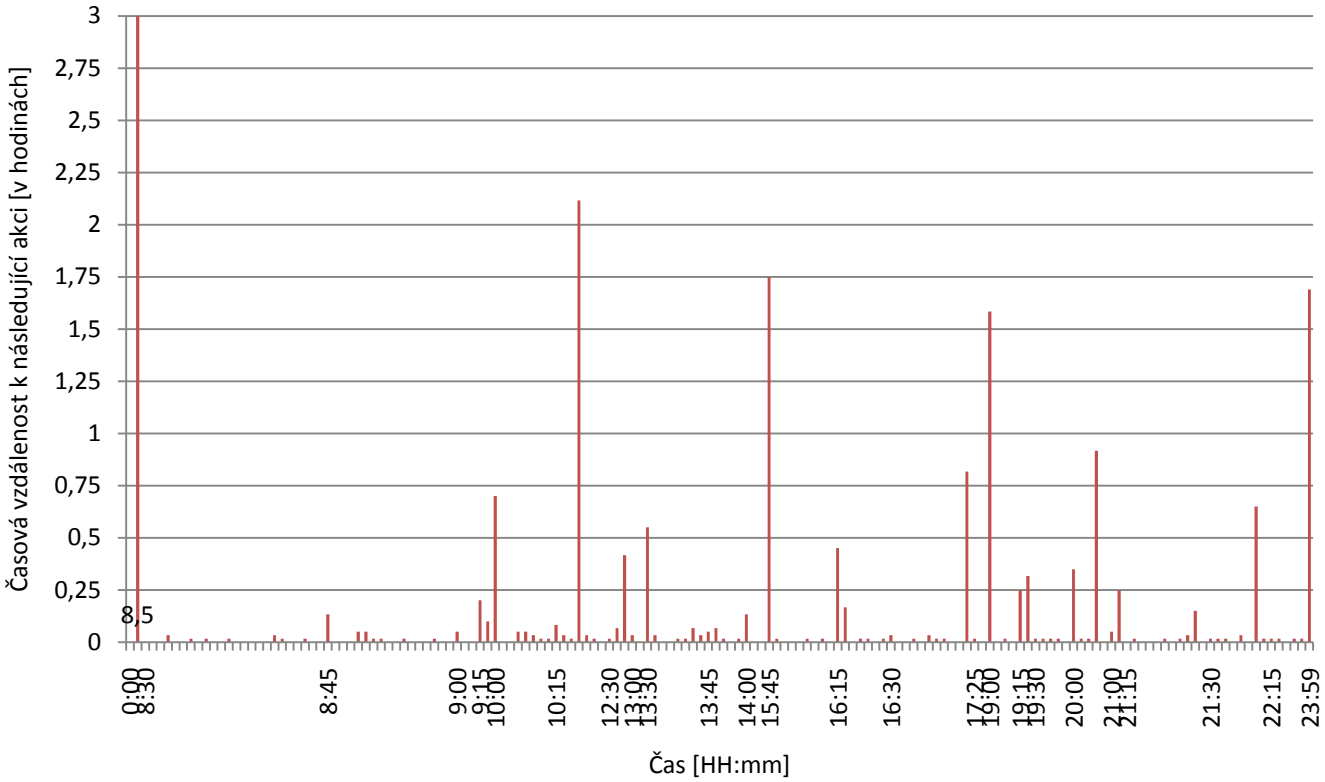
Na tomto místě bylo nutné učinit rozhodnutí, jak zvolit strukturu intervalů a jak volit jejich délku. Jak dlouhé volit ty nejkratší z nich a jak naopak ty nejdelší. Bylo rozhodnuto rozdělit den do čtyř druhů intervalů: nejkratší 15 minutový, dále 30

minutový, 45 minutový a nejdelší 60 minut. Intervaly tedy začínají vždy ve čtvrt, půl, tři čtvrtě, nebo celou hodinu. V podstatě to vypadá tak, že knihovna si nejprve zjistí z logu



Obr. 4 Schéma zpracování vstupních dat (logu)

všechny časy akcí. Zjistí čas první události a určí, do které čtvrthodiny patří, tu vezme jako referenční. V dalším postupu zjistí časy následujících akcí. Jestliže jich je dostatečný počet do nejkratšího možného typu intervalu, tedy patnáctiminutového, vytvoří tento interval s počátkem v referenčním čase a pokračuje na další akci s časem za tímto intervalem. Pokud ne, rozšíří pole působnosti na druhý typ intervalu (třicetiminutového) a v případě, že se do třiceti minut vykonalo dostatek událostí, vytvoří třicetiminutový interval začínající v referenčním čase. Jestliže ani do třiceti minut neproběhlo dostatečné množství akcí, pokračuje na další interval. V případě neúspěchu na nejdelší šedesátiminutový interval. Po vytvoření intervalu se program vrátí na akci začínající v čase za koncem tohoto intervalu. Znovu se snaží tvořit další intervaly a to opět od těch nejkratších. Přiřazení intervalů jednotlivým časům demonstruje tabulka 1 na následující straně. Tento proces odpovídá vynesení jednotlivých časů na časovou osu a následné nalezení nejhustějších částí s nejkratším trváním patnáct minut. Tedy jakýsi histogram událostí (viz graf na obr. 5).



Obr. 5 Graf vzdálenosti akce k následující akci

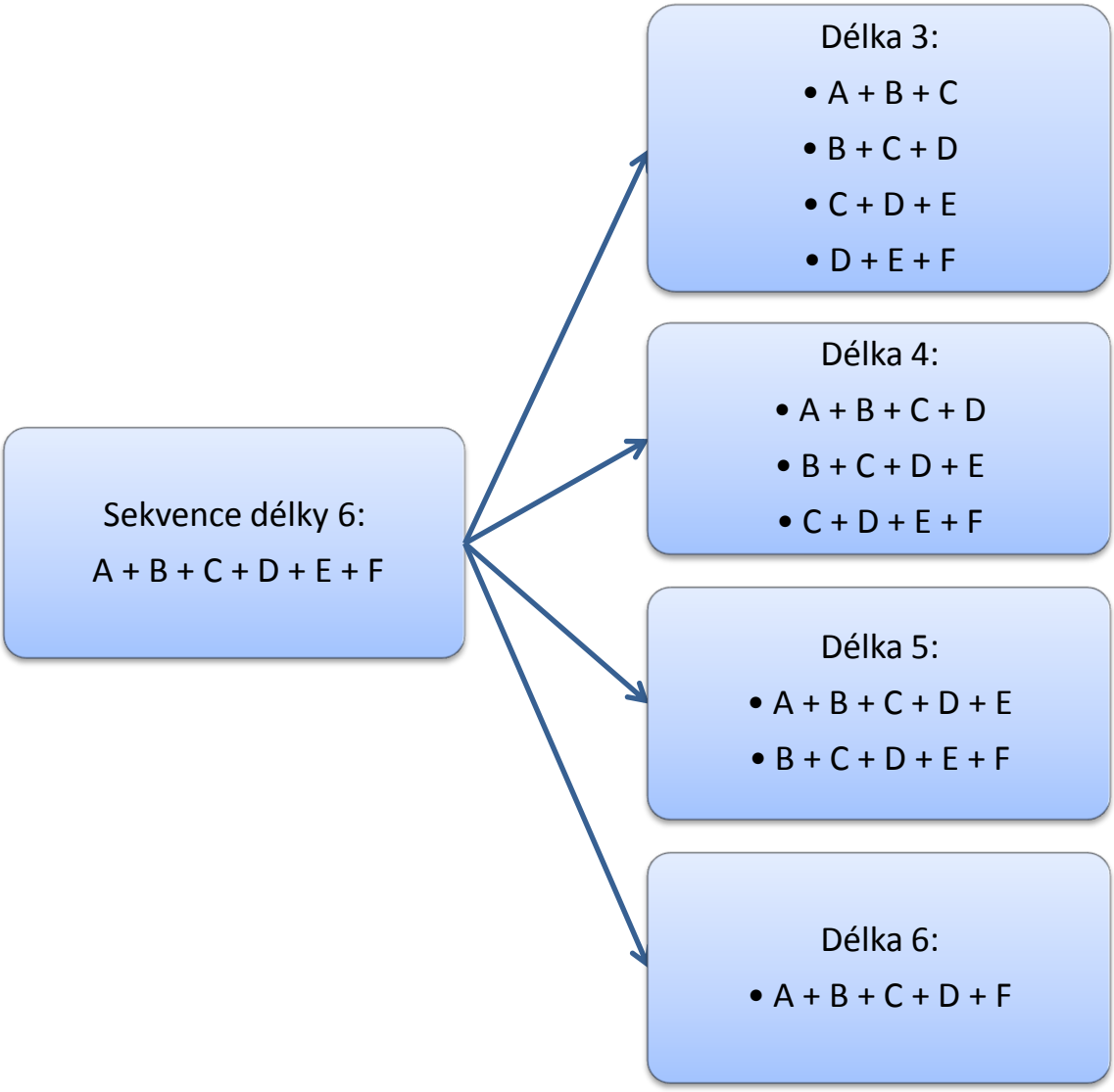
Tabulka 1 Příklad přiřazení intervalu k času

čas		konec intervalu	čas		konec intervalu	čas		konec intervalu
		8:30	12:33	→	13:15		→	18:30
8:30	→	8:45	12:58	→	13:15		→	19:00
8:32	→	8:45	13:00	→	13:15	19:00	→	19:15
8:33	→	8:45	13:33	→	13:45	19:01	→	19:15
8:34	→	8:45	13:35	→	13:45	19:16	→	19:45
8:35	→	8:45	13:36	→	13:45	19:35	→	19:45
8:37	→	8:45	13:37	→	13:45	19:36	→	19:45
8:39	→	8:45	13:41	→	13:45	19:37	→	19:45
8:47	→	9:00	13:43	→	13:45	19:38	→	19:45
8:50	→	9:00	13:46	→	14:00	19:39	→	19:45
8:53	→	9:00	13:50	→	14:00	20:00	→	20:15
8:54	→	9:00	13:51	→	14:00	20:01	→	20:15
8:55	→	9:00	13:52	→	14:00	20:02	→	20:15
8:56	→	9:00	14:00	→	14:15	20:57	→	21:15
8:57	→	9:00		→	15:15	21:00	→	21:15
9:00	→	9:15		→	15:45	21:15	→	21:30
9:12	→	9:15	15:45	→	16:00	21:16	→	21:30
9:18	→	10:00	15:46	→	16:00	21:17	→	21:30
10:00	→	10:15	15:47	→	16:00	21:18	→	21:30
10:03	→	10:15	15:48	→	16:00	21:20	→	21:30
10:06	→	10:15	16:15	→	16:30	21:29	→	21:30
10:08	→	10:15	16:25	→	16:30	21:30	→	21:45
10:09	→	10:15	16:26	→	16:30	21:31	→	21:45
10:10	→	10:15	16:27	→	16:30	21:32	→	21:45
10:15	→	10:30	16:28	→	16:30	21:32	→	21:45
10:17	→	10:30	16:30	→	16:45	21:34	→	21:45
10:18	→	10:30	16:31	→	16:45	22:13	→	23:59
	→	11:30	16:33	→	16:45	22:14	→	23:59
	→	12:15	16:34	→	16:45	22:15	→	23:59
12:25	→	12:30	16:35	→	16:45	22:16	→	23:59
12:27	→	12:30		→	17:15	22:17	→	23:59
12:28	→	12:30	17:24	→	17:30	22:18	→	23:59
12:29	→	12:30	17:25	→	17:30			

V druhém kroku knihovna jednak data roztřídí podle místností, ve kterých se akce vykonal a dále v rámci místnosti zařadí akce dle času, ve kterých se udály. Knihovna je napsána dostatečně obecně, takže není překážkou kdykoliv přidat novou akci nebo rozšířit pole působnosti na další lokace. Vznikne tak poměrně složitá, ale logická struktura, kde každé akci je přiřazeno konkrétní místo v této struktuře na základě pozice a času, respektive časového intervalu, do kterého spadá. Současně je zaznamenána také četnost události v rámci konkrétní pozice a daného časového intervalu. Celá struktura je detailně popsána v kapitole 5.2 a zobrazena na obr. 6.

Za třetí program vyhledá v záznamu neboli logu sekvence akcí, které spolu souvisí, neboli jsou vykonávány v přímém sledu za sebou a to v relativně krátkém časovém okamžiku po sobě. Na tomto místě bylo nutné udělat dvě rozhodnutí při návrhu. Jednak určit co ještě je a co už není posloupností akcí tak, aby to mělo reálný význam pro predikci, a současně se rozhodnout, jak dlouhý nebo krátký má být časový interval mezi každými dvěma akcemi po sobě, případně jak definovat sekvenci akcí. Jako minimální délka posloupnosti byl zvolen sled minimálně tří akcí. Jedna akce netvoří sekvenci, protože nelze určit souvislost s dalšími akcemi v sekvenci. U dvou akcí už lze zaznamenat jistou souvislost mezi akcemi. Výsledný počet akcí o minimální velikosti dva by byl ale příliš vysoký a v podstatě by uživateli nepomohl. Uživatel by musel sekvenci stejně potvrdit, stejně jako by potvrdil / vykonal pouze druhou akci. Vzrostla by doba výpočtu a reálně by se nijak výsledné predikci nepomohlo, spíše by byla méně přesná. Proto byly zvoleny tři akce jako minimální počet akcí, které tvoří sekvenci. Rovněž je nutné se rozhodnout pro stanovení maximální doby vždy mezi dvěma po sobě následujícími akcemi tak, aby tvořily sekvenci. V reálné aplikaci by bylo nejpřesnější po nějaké době přehodnotit úspěšnost předpovědi a časový interval optimalizovat dle konkrétního uživatele a výsledků. Protože nebylo možné návrh opravdu reálně a dlouhodobě otestovat, bylo rozhodnuto zvolit pro aplikaci dobu mezi dvěma po sobě následujícími akcemi jako interval deseti minut. Je to kompromis mezi trváním nejkratšího intervalu, tedy 15 minut, a zároveň odhad průměrné délky činnosti akce připadající v úvahu jako součást sekvence. U sekvence delší než tři program zároveň uloží i podsekvence s minimální délkou právě tři. Tedy najdeme-li například sekvenci o délce 6 akcí, nezaznamenáme jenom tuto jednu dlouhou sekvenci, nýbrž

současně všechny kratší podsekvence o délce tři, čtyři a pět. Pro lepší představu pomůže následující obrázek:

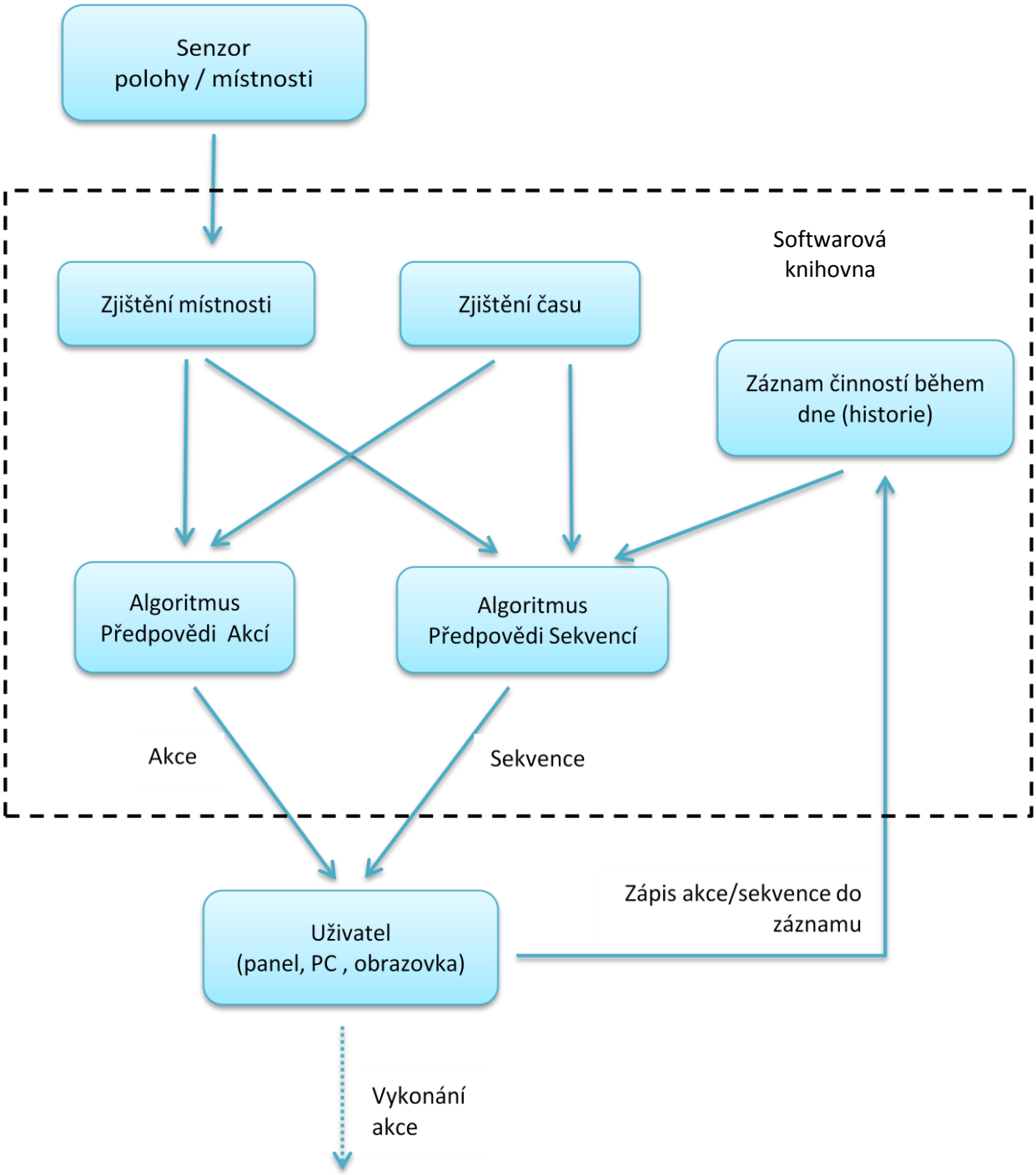


Obr. 6 Příklad vytvořeni podsekvencí z delší sekvence

Program také zaznamená četnost všech těchto sekvencí a stejně tak podobnost s jinými sekvencemi a vše je uloženo do struktury, která má ovšem již značně jednodušší architekturu oproti struktuře uchovávající samotné akce. (podrobněji o tomto v kapitole Implementace v oddílu 5.2).

4.3. Predikce

Pokud jsou dokončeny úspěšně předešlé části, program je připraven k použití. Algoritmus jeho činnosti je následující. Vždy nejprve zjistí v jaké pozici, respektive v jaké místnosti se uživatel nachází. K tomu se využijí v bytě zabudované senzory, které do řídící jednotky předají informaci udávající polohu a ta se dále zašle až do modulu. Senzory nejsou součástí knihovny, ta pracuje s již přijatými informacemi, tedy předpokládá, že komunikaci se spotřebiči, které jsou v domu, nebo bytě, obstarává jiná součást systému. Současně si knihovna dále zjistí aktuální čas a nalezne ze seznamu interval, který mu náleží. Na základě těchto údajů vybere ze struktury akcí tři nejvíce vyhovující akce pro následující uživatelovu činnost (existují-li) a zobrazí je. Spolu s tím se knihovna také snaží najít vhodnou sekvenci akcí ze struktury sekvencí a zbytek, který ještě nebyl proveden, nabídnout uživateli. Všechny akce, které uživatel během dne učiní, se samozřejmě rovněž zaznamenávají. Pro vyhledání sekvence se tento záznam dosud vykonaných akcí využívá, aby bylo možné ve struktuře nalézt podobnou sekvenci. Na konci dne, případně kdykoliv během dne, se tato historie dne zapíše do logu, kde poslouží ke zlepšení predikce. K znázornění předešlého viz obr. 7. Formát dat v logu má konkrétní tvar a blíže je popsán v podkapitole 5.1.



Obr. 7 Postup predikce pro konkrétní situaci

5. Implementace

Cílem této části je popsat konkrétně způsob implementace softwarové knihovny v jazyce C# pod platformou .NET. Pro implementaci celé knihovny jsem použil integrované vývojové prostředí Microsoft Visual Studio 2010 Express, které je pro tento projekt zcela dostačující a je rovněž zcela zdarma dostupné. Programovací jazyk C# byl zvolen z toho důvodu, že celý systém, do kterého by měla být knihovna zakomponována, je taktéž psán v jazyku C#. Při implementaci bylo využíváno online dokumentace Microsoft .NET Framework [5] i jiných příkladových kódů v jazyku C# [6].

Celá knihovna je patřičně zdokumentována i v zdrojovém kódu.

5.1. Záznam - log

Jak bylo řečeno v předešlé kapitole Návrh, textový soubor obsahující záznamy typu: čas, pozici a činnost, která v tomto čase a na tomto místě byla vykonána, je nazýván log. Protože log je pokaždé při prvotním startu knihovny znova načítán a zpracováván, je nutné uchovávat data logu v určité podobě za účelem automatického zpracování. Formát logu je tedy přesně daný. Každá akce spolu s upřesňujícími údaji, tedy časem a místností, je zaznamenána vždy na jeden řádek. Zároveň jsou odděleny jednotlivé dny a zapsány i jejich data.

Obecně má jeden řádek logu tento formát:

```
čas/pozice[akce];
```

Čas je ve tvaru HH:mm, kde HH znamená 24hodinový formát času, mm jsou minuty (např. 08:25). Informace o dni a jeho datu má následující tvar:

```
#denVtýdnu_datum
```

Datum má tvar ddMMyyyy. První dvě místa znamenají pořadí dne v měsíci, MM udává pořadí měsíce v roce a konečně yyyy je letopočet.

Na ukázku několik zaznamenaných akcí ze dne pondělí 28. února 2011:

```
#monday_28022011
0830/bedroom[alarm];
0832/bedroom[veiling_up];
0834/bedroom[window_open];
0837/bathroom[light_on];
0850/bathroom[light_off];
0853/livingroom[veiling_up];
0855/livingroom[window_open];
0856/livingroom[tv_ct24];
1000/livingroom[tv_off];
...
```

Tolik formát logu. Převedeme-li výše prezentovaný úryvek do běžných vět, tak obdržíme toto: uživatel se vzbudil, respektive mu zazvonil budík, v půl deváté ráno ve své ložnici, o dvě minuty později vytáhl ve stejné místnosti závěsy a o další chvíli později otevřel okno. Dále pokračoval do koupelny, kde si rozsvítil, a po delší době z ní zase nejspíše po vykonání ranní hygieny vyšel, protože zhasnul světlo a následně se objevil v obývacím pokoji. Zde také vytáhl závěsy, nebo rolety a otevřel okno. Po té začal sledovat televizi a to konkrétně stanici ČT24, načež televizi v deset hodin vypnul. Záznam pokračuje dále až na konec dne a pak obdobným stylem i další dny.

Třídy a metody jsou napsány zcela obecně, takže jim nečiní problém, přibude-li nová akce nebo místnost. Nicméně ve svém pracovním logu jsem z pochopitelných důvodů používal konečný a omezený počet jak akcí, tak místností. Seznam akcí stejně jako výčet místností jsem přizpůsobil 3D modelu bytu, který používá skupina Nature Inspired Technologies Group na katedře kybernetiky ve své Aplikaci pro řízení domácího prostředí. V programovacím kódu jsem používal výhradně anglického jazyka, proto i následující výčet bude v angličtině:

Místnosti:

- bedroom
- bathroom
- livingroom
- cabinet

Akce:

- alarm
- veiling_up
- window_open
- light_on
- light_off
- tv_ct24
- tv_off
- window_closed
- radio_cro
- radio_off
- tv_ct1
- fan_on
- fan_off
- tv_nova
- tv_prima
- veiling_down

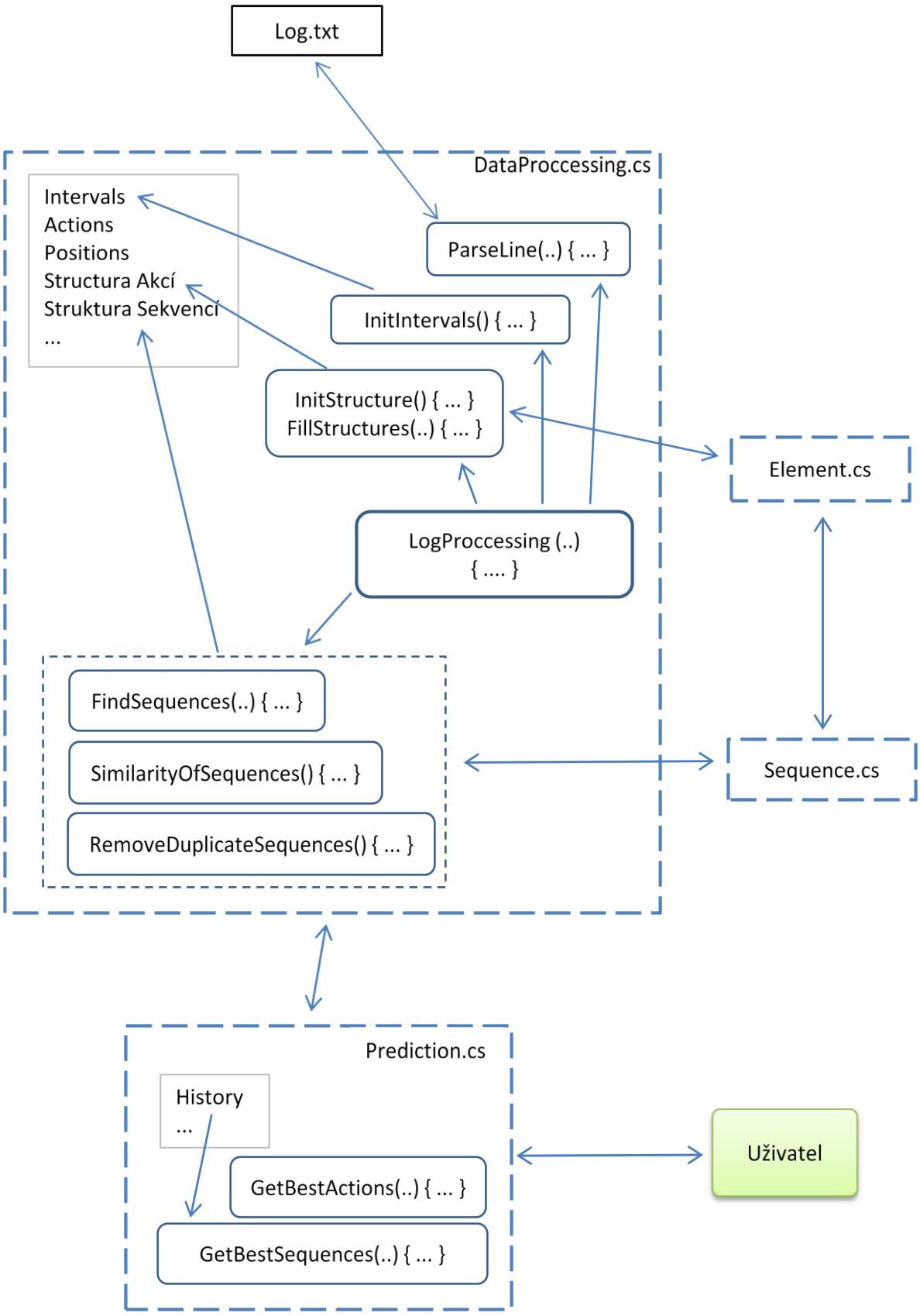
Konec celého logu musí být označen tagem `>>end`.

5.2. Třídy

Výsledná knihovna, kterou jsem pojmenoval ActionsPrediction, obsahuje pět hlavních tříd:

- `Atom`
- `Element`
- `Sequence`
- `DataProccessing`
- `Prediction`

Nyní stručně k jednotlivým třídám a jejich významům. Celé fungování knihovny znázorňuje obr. 8.



Obr. 8 Fungování knihovny

5.2.1. Třída Atom

Je bázovou třídou, od které jsou odvozeny dvě následující třídy: `Element` a `Sequence`. Její přetížený konstruktor má 4 parametry:

- `action : string`
- `position : string`
- `Time: DateTime`.
- `Done : bool` údaj zda akce byla vykonána či je pouze predikována

Třída slouží také pro jednotné zobrazení vykonaných nebo predikovaných akcí v aplikaci Demo (viz kapitola 7. Demo).

5.2.2. Třída Element

Třída `Element` dědí od třídy `Atom`. Reprezentuje vždy jednu konkrétní akci spolu s časem a místem, kde se udála. Pro vytvoření nové akce typu `Element` se používá přetížený konstruktor s parametry:

- `action : string`
- `position : string`
- `Time: DateTime`.

Volá se vždy, při vkládání nové akce například do struktury akcí anebo při tvoření sekvencí akcí pro tvorbu jednotlivých prvků. Třída `Element` používá pro zajištění základních údajů vlastnosti (properties):

- `Action : string`,
- `Position : string`
- `Time : DateTime`.

Dále implementuje metody:

- `bool IsSame(Element e)`, která porovnáním hodnot `Action` a `Position` zjistí, zda jsou dva prvky totožné.

- Přetěžuje metodu `ToString()` pro výpis `Elementu` ve tvaru `action[position]`

5.2.3. Třída Sequence

Reprezentuje posloupnost nebo sekvenci akcí, tak jak byla popsána v kapitole 4. Návrh, tedy sekvenci akcí o minimální délce tří akcí. Zároveň musí být časová vzdálenost dvou po sobě následujících akcí do deseti minut. Přetížený konstruktor této třídy přijímá při inicializaci pět parametrů:

- `List<Element> sequence` je seznam typu `Element`, který obsahuje jednotlivé akce typu `Element`, tak jak budou vloženy do sekvence
- `DateTime start` je začátek sekvence v typu `DateTime`
- `DateTime end` je konec sekvence v typu `DateTime`
- `int occurance` je počet výskytů této sekvence
- `int occuranceOfSimilarity` je počet výskytů podobných sekvencí

Třída implementuje několik vlastností (properties), které jsou velmi podobné parametrům v konstruktoru:

- `public int Count` udává délku sekvence
- `public List<Element> ElementsOfSequence` je seznam akcí typu `Element` obsažených v sekvenci
- `public int Occurance` udává počet výskytů této sekvence
- `public int Similar` počet výskytů podobných sekvencí (podobnost sekvencí viz níže)
- `public DateTime StartOfSeq` začátek sekvence
- `public DateTime EndOfSeq` konec sekvence
- `public Element this[int index]` deklarace indexů, která umožňuje přístup na jednotlivé akce typu Element v sekvence přímo pomocí indexů

5.2.4. Třída DataProccessing

Obsahově největší třída. Její funkcí je zpracovat textový soubor uchovávající záznam o činnosti uživatele, tedy log, a inicializovat celou strukturu akcí a seznam sekvencí spolu s jejich doplňkovými informacemi. Struktura akcí má relativně složitou koncepci. Jedná se o několik různých do sebe vnořených generických tříd. Nejlépe vystihuje celou strukturu obr. 9. Vnější vrstvou je List, který obsahuje pro každou místnost jednu kolekci typu `SortedDictionary`. Ta používá jako klíče (`Keys`) časové intervaly, tak jak byly vytvořeny na začátku startu knihovny a jak byly přizpůsobeny hustotě akcí na časové ose (viz kapitola 4. Návrh). Jako hodnoty (`Values`) jsou ke klíčům přiřazeny další kolekce typu `Dictionary`. Každá tato kolekce je naplněna dvojicemi `key – value.` Jako `key` slouží všechny akce. Ke každému klíči neboli akci je přiřazena jako hodnota trojrozměrné pole typu `Integer`. V tomto poli jsou zaznamenány pro každou akci, pro každý časový interval a pro každou místnost výskyty těchto akcí. Tedy kolikrát se v tento interval a na tomto místě v minulosti udály. Na první pozici pole je údaj o počtu výskytů v posledních sedmi dnech, na druhém indexu v posledních 30 dnech a na poslední pozici celkový výskyt. K naplnění těchto údajů dochází při prvotním startu knihovny, respektive celého programu.

Hlavní metodou této třídy je metoda:

- `public void LogProcessing(string filename, DateTime today)`

Ta přijímá jako parametr jednak adresář obsahující soubor log a jednak datum dne, ke kterému je zpracování logu vykonáváno. Metoda je volána jako zcela první a zajišťuje, jak už název napovídá, zpracování souboru log. V jejím těle jsou v několika různých `for`- případně `foreach`-cyklech volány další pomocné metody obsažené v třídě `DataProccessing`. Nejprve metody pro zjištění repertoáru akcí a místností z logu. Spolu s nimi rovněž metoda

- `private void InitIntervals(List<DateTime> x_Times)`

starající se o již zmíněné vygenerování časových intervalů. Jako další se inicializuje struktura akcí pro následný zápis hodnot. Toto zařídí metoda

- `private void InitStructure()`.

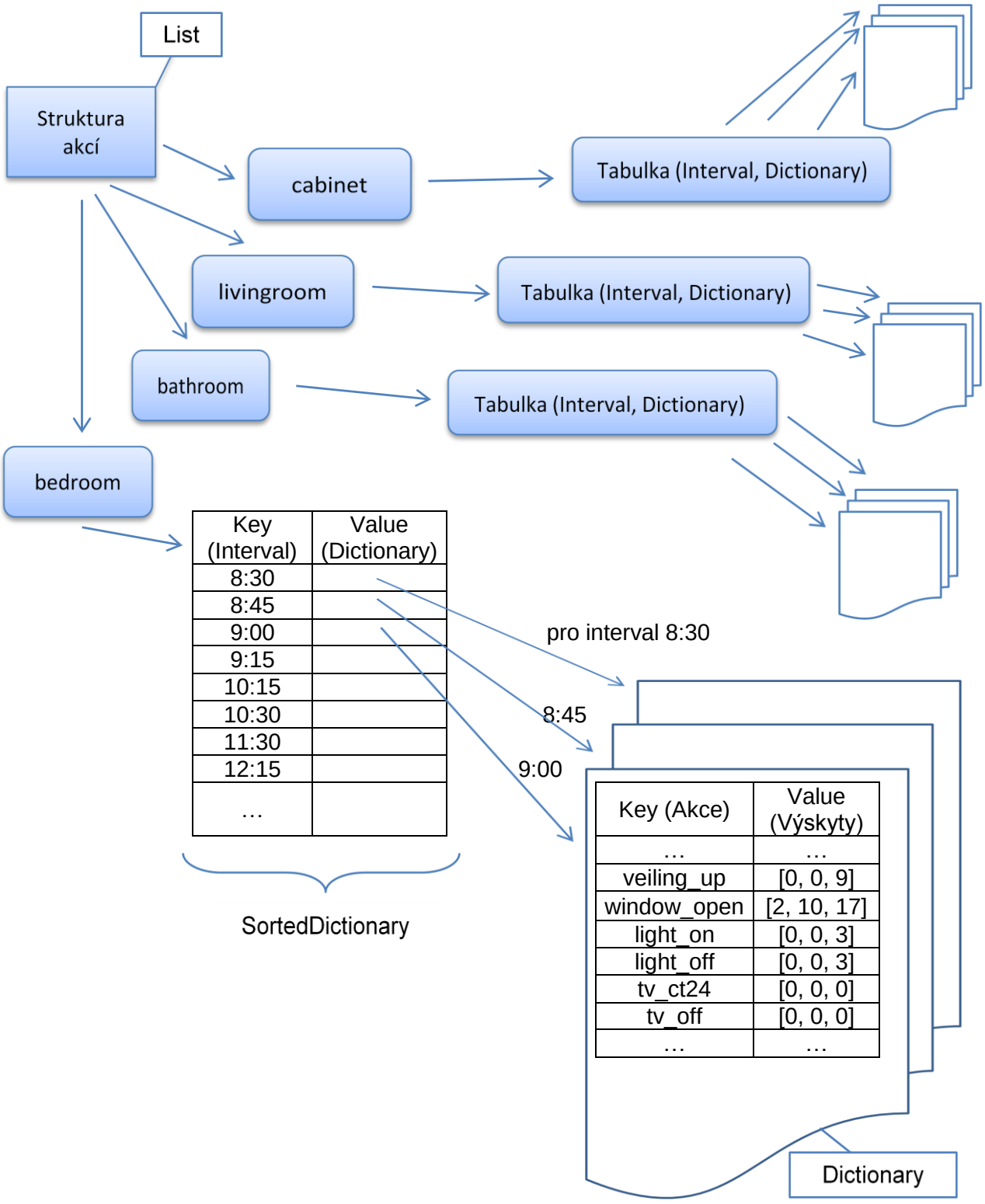
Nyní je možné do takto připravené struktury pomocí `for`-cyklu zapisovat postupně hodnoty. Cyklus přečte vždy jeden řádek z textového souboru log. Ten má přesně stanovený formát a pomocí metody

- `public string[] ParseLine(string line)`

vrátí v poli typu `string` čas, pozici a akci z načteného řádku. Známe-li tyto údaje, může další metoda

- `private void FillStructures(DateTime Time, string action, string position)`

přesně přistoupit do struktury akcí na danou pozici. Zde vyhledá pomocí intervalu-klíče hodnotu reprezentovanou generickým typem `Dictionary`, ve které za pomoci akce-klíče nalezne pole, u kterého zvýší patřičný výskyt. Zapisování se zopakuje pro všechny řádky v logu, které obsahují akci. Výsledkem bude naplněná struktura akcí. Celá struktura je zachycena na obr. 9.



Obr. 9 Struktura akcí v třídě DataProccessing

Po této části se začne taktéž z logu za pomoci metody

- `private void FindSequences(string path)`

tvořit seznam sekvencí. Jak již bylo řečeno v kapitole 4.2, jako sekvenci je brána posloupnost minimálně tří akcí a zároveň musí být od sebe každé dvě následující akce časově vzdáleny do deseti minut. Takto vytvoříme seznam. Je ale pravděpodobné, že se v něm budou vyskytovat některé sekvence, které budou zcela stejné. Exkluzivitu sekvencí zaručí metoda

- `private void RemoveDuplicateSequences()`,

která ponechá vždy jeden výskyt sekvence. Zároveň u sekvencí, které byly v původním seznamu vícekrát, zvýší vlastnost `Occurance`, tedy počet výskytů této sekvence v logu.

Poslední hlavní částí je metoda

- `private void SimilarityOfSequences()`

detekující podobnost mezi jedinečnými sekvencemi, které zůstaly v seznamu. Stanoví u každé z nich vlastnost `Similarity` (podobnost) určující, s kolika sekvencemi si je tato sekvence podobná.

Aby mohly být dvě sekvence podobné, stanovil jsem hranici podobnosti sekvencí následovně:

a) Při stejném počtu akcí v sekvencích, tedy při stejné délce sekvencí, zároveň délce větší než tři a zároveň ne větší než šest, maximálně jednu odlišnost mezi sekvencemi.
   Např.: Sekvence ABCD je podobná AXBC i AXCD. Tedy je možné nahradit jeden prvek jiným, nebo vložit mezi prvky další prvek.
   Při délce větší než 6 jsou tolerovány dvě odlišnosti.

b) Má-li jedna sekvence o jeden prvek více, nebo méně, je tolerován jeden další posun / odlišnost v sekvencích. Vždy porovnávám z pohledu kratší sekvence.
   Např. ABC je podobné ABCX, XABC, AXBC, AXCD apod.
   ABCD je podobná sekvenci XABCD, ABCDX, AXBCD, AXCDE apod.

c) Při rozdílu dvou prvků mezi porovnávanými sekvencemi musí být menší sekvence podmnožinou té větší.
Např. ABC je podobné ABCXY, XABCY,XYABC.

5.2.5. Třída Prediction

Zajišťuje samotnou predikci. Konstruktor třídy získá jako parametr proměnou typu `DataProccessing`, která obsahuje všechna potřebná data. Zároveň si také vede globální proměnou `public List<Element> History`, do které si zaznamenává činnosti vykonané ten den uživatelem. Specifika třídy DataProccessing byla popsána v předcházející části. Třída `Prediction` implementuje pouze dvě metody:

- `public List<Element> GetBestActions(DateTime TimeAction, string position)` určí pro daný čas a pozici tři nejlepší akce (existují-li) dle struktury akcí, tak jak byla vytvořena na základě logu

- `public List<Sequence> GetBestSequences(DateTime time)` vrátí naopak dvě nejpravděpodobnější sekvence. K nalezení nejvhodnějších sekvencí využívá výše zmíněný seznam `History`, ze kterého načte několik posledních akcí, na základě kterých prohledává list sekvencí a zaznamenává ty, které se nejvíce shodují.

Každá nová vykonaná akce nebo celá sekvence akcí se přidá vždy na konec `History`. Seznam `History` navíc slouží k aktualizaci dalších dat do logu. Zapsání může být provedeno automaticky buď v konkrétní dobu v noci, nebo manuálně v libovolný čas. Metody třídy `Prediction` jsou volány po každém vykonání akce, aby navrhly další akce a sekvence.

5.3. Testování

Celá predikce byla zkoušena na několika automaticky vygenerovaných textových souborech (log). Ty simulují činnost člověka v inteligentním domově po dobu 30 nebo

60 dnů, tedy jednoho či dvou měsíců. Pro vygenerování logů byla vytvořena jednoduchá třída `LogGenerator`, která logy automaticky vygenerovala. Jako základ pro tvorbu náhodného logu sloužil ručně vytvořený log, který měl několik dnů. Metody třídy `LogGenerator` tento log použily jako vstup a náhodně („pseudonáhodně“) posunuly časy v intervalu plus minus deset minut. Načtené dny z původního ručně vytvořeného logu byly rozděleny na tři časová pásma. Zjednodušeně řečeno na dopoledne, poledne a odpoledne, tedy do 10:30, od půl jedenácté do půl čtvrté a od půl čtvrté do půlnoci. Z prvního a druhého časového pásma vybraly náhodně deset akcí, ke kterým bylo podobně přiřazeno deset náhodně vybraných časů náležících tomuto časovému pásmu. Z posledního časového pásma bylo vybráno dvacet akcí a k nim dvacet časů. Takto vznikl jeden den sestávající ze čtyřiceti akcí. Stejným postupem byly vytvořeny logy o třiceti a šedesáti dnech a to vždy šestkrát. Celkově tedy dvanáct různých logů dvou různých délek, náhodně vytvořených.

6. Manuál

Vytvořená knihovna sama o sobě přímo žádné akce nevykonává. Pro její uvedení do provozu je potřeba několika kroků. Jako první je potřeba inicializovat proměnou typu `DataProccessing` (viz kapitola 5 - Implementace). Následně nad ní zavolat metodu `LogProccessing` a předat jí dva parametry. Prvním je adresa místa, kde se nachází textový soubor log. Druhým parametrem je datum. Log je potřeba převést do formátu, jak popisuje kapitola 5.1. Není-li metodě předán druhý parametr, sama si doplní skutečné aktuální datum. Dále je potřeba inicializovat proměnnou třídy `Prediction` (také viz kapitola Implementace). Nad ní se vždy po vykonání akce volají metody `GetBestActions (…)` a `GetBestSequences(…)`, které vracejí `List<Element>`, respektive `List<Sequence>`, jejichž výstupem jsou navržené akce, respektive sekvence akcí. Zároveň je do globální proměnné `History` nad proměnnou typu `Prediction` po každé vykonané akci potřeba tuto akci, případně sekvenci, uložit. `History` je typu `List<Element>`. Proto je potřeba akce před přidáním do tohoto seznamu převést do typu třídy `Element`. To se uskuteční velmi snadno inicializováním nové proměnné pomocí konstruktoru `Element (…)`, který obsahuje tři parametry. Prvním je název vykonaná akce v typu `String`, druhým místnost taktéž typu `String` a posledním je čas vykonání akce v typu `DateTime`. Jednotlivé části jsou detailněji popsány v 5. kapitole - Implementace.

Výše popsané lze shrnout do několika bodů:

- Zadání adresáře s log.txt.
- Inicializace proměnné typu DataProccessing
- Zavolání metody LogProccessing s parametry adresáře logu a „dnešním“ datem nad touto proměnnou.
- Inicializace proměnné typu Prediction
- Volání metod GetBestActions (…) a GetBestSequences(…) s příslušnými parametry.
- Ukládání vykonaných akcí v typu Element do Listu History nad proměnnou typu Prediction.

V následující kapitole je popsána funkce demonstrační aplikace implementující výše zmíněné. Ovládání je uskutečňováno přes grafické uživatelské prostředí a není nutné nic dále programovat.

7. Demonstrační aplikace

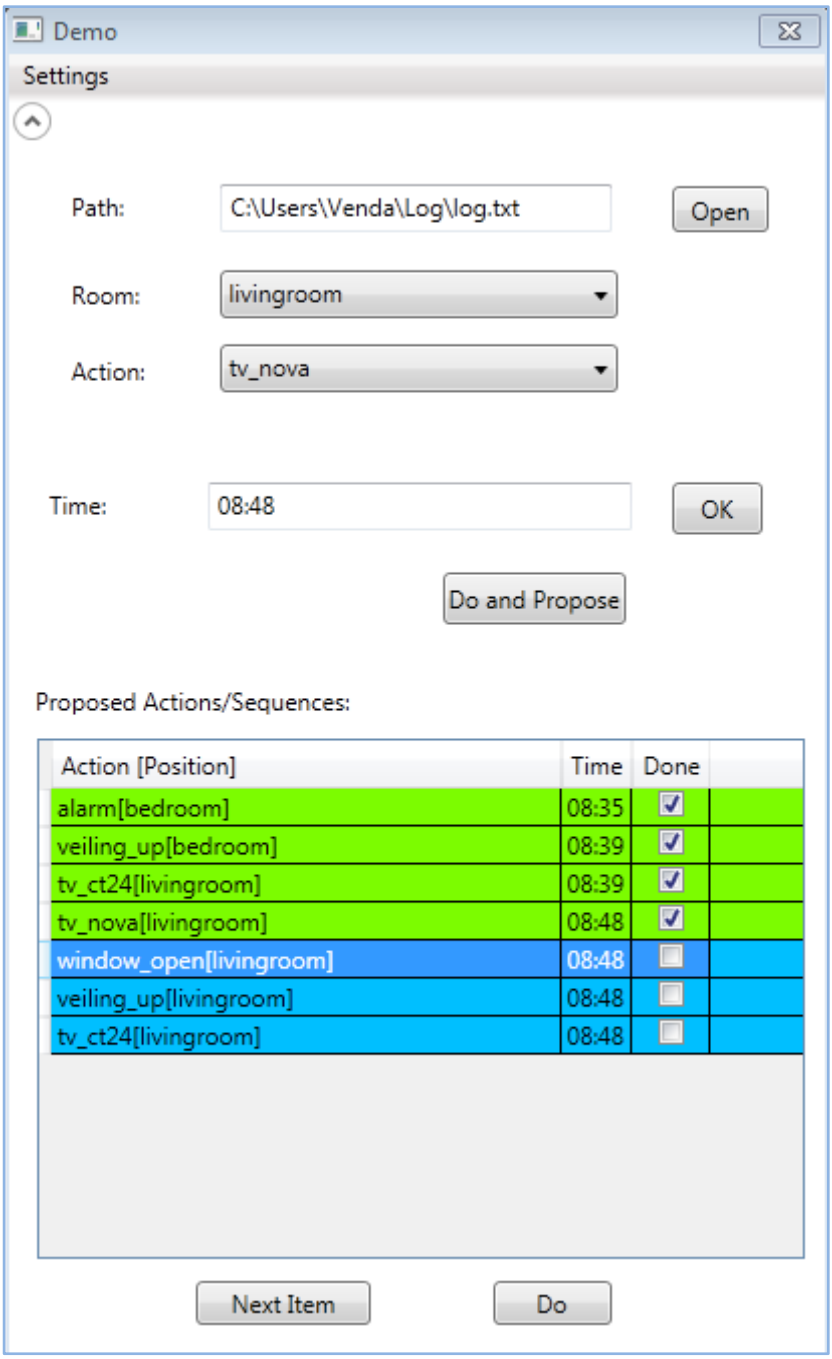
Demonstrační aplikace je spustitelnou aplikací s grafickým uživatelským rozhraním sloužící především k ukázce činnosti knihovny. Demo je obsaženo na přiloženém CD v adresáři Actions Prediction. Spustit ho lze pomocí souboru Demo přímo ve složce Actions Prediction. Vytvoření aplikace nebylo hlavním cílem této bakalářské práce, ale bez ní by bylo velmi obtížné a zdlouhavé činnosti knihovny vyzkoušet, nebo i dokonce předvést. Grafické rozhraní bylo záměrně navrženo co nejjednodušeji, ale zároveň aby na něm bylo možné simulovat vše potřebné. Demo simuluje činnost „jednoduchého“ uživatele v domácnosti. Pomocí tlačítek a seznamů „vykonáme“ v určitý čas konkrétní činnost na konkrétním místě tak, jako bychom jí vykonali ve skutečném domově.

Nejprve popis ovládání dema. Vzhled grafického rozhraní aplikace je zobrazen na obr. 10 na straně 43. V hlavním okně je několika tlačítek, comboboxů a tabulka pro zobrazení, kterou zastupuje komponenta `DataGrid`. Jako první nastavíme pomocí dialogu přes tlačítko „Open“ adresář, ve kterém se nachází log se vstupními údaji. Ty knihovna potřebuje načíst jako první. K tomu slouží přiložené textové soubory log A až F. Po té se naplní první a druhý combobox s místnostmi a možnými akcemi. Pod nimi se nachází umístěno textové pole, do kterého se zapisuje čas, v kterém se má událost uskutečnit. Čas musí být v 24hodinovém formátu ve tvaru HH:mm, například 08:49. Pro potvrzení času slouží tlačítko „OK“ vedle textového pole. V případě jeho nepotvrzení se v dalších krocích i přesto použije čas zadaný v poli Time. Jakmile jsou nastaveny místnost, akce a čas, můžeme kliknutím na tlačítko „Do and Propose“ spustit algoritmus predikce.

Níže v okně se zaplní tabulka navrhovanými akcemi, nebo sekvencemi akcí. Na každém řádku je jedna akce nebo sekvence akcí. Tabulka má tři sloupečky. V prvním sloupci s názvem Action [Position] je informace, jaká akce a v jaké místnosti byla vykonána. V druhém sloupečku (Time) je zobrazen čas a v posledním (Done) zda akce již byla vykonána uživatelem nebo zda se jedná o navrhovanou akci. To je navíc rozlišeno barevně. Modře jsou vyznačeny řádky s již vykonanými akcemi v pořadí, v jakém následovaly za sebou. Za nimi zeleně navrhované jednotlivé akce. Sekvence akcí jsou zobrazeny jako několik akcí za sebou spojené symbolem „+“. Vybráním jedné položky z návrhu a potvrzením tlačítkem „Do“ spustíme znova predikci a zobrazí se

nové navrhované akce, nebo sekvence. Je dobré změnit před kliknutím na „Do“ čas výše v okně. Tlačítko „Next Item“ slouží k označení dalšího prvku v tabulce nad ním. Toto tlačítko je zde pro možnost ovládání grafického uživatelského prostředí tělesně postiženým člověkem, nebo starší osobou, tak aby bylo použito nejmenšího počtu pohybů a tlačítek. Pro ovládání by pak stačila pouze dvě tlačítka: „Next Item“ a „Do“. Chceme-li vykonat jinou akci, než kterou nabízí tabulka, nebo na jiném místě, vybereme je znovu výše pomocí comboboxů a určením času.

Menu Settings (nastavení) v horní liště obsahuje dvě položky. „Add Day To Log“ a „Today“. Při běhu programu, tak jak „konáme“, se tyto akce zapisují do seznamu, který, jak bylo řečeno v páté kapitole Implementace, slouží jednak pro vyhledávání sekvencí a jednak pro zaktualizování logu. A právě pro připsání těchto nově vykonaných akcí neboli záznamu „celého dne“ do logu slouží první tlačítko v menu Settings. Druhá položka má funkci nastavení data „dnešního“ dne, ke kterému se bude predikce vztahovat. Jestliže ho nenastaví uživatel, program automaticky vezme skutečné dnešní datum. V aplikaci se jako „dnešní“ datum defaultně nastaví den po konci šedesátidenních logů, tedy 30. březen 2011. Po kliknutí na nabídku „Today“, vyskočí dialogové okno s kalendářem. V něm uživatel označí patřičný den a tlačítkem „Confirm“ se volba potvrdí.



Obr. 10 Demo - Grafické uživatelské rozhraní

8. Závěr

Cílem bakalářské práce bylo seznámit se s možnostmi predikce akcí uživatele v domácích inteligentních řídicích systémech neboli inteligentních domech a současně navrhnout metody poskytující vhodnou predikci. Hlavním úkolem bylo vytvořit softwarovou knihovnu, která bude navržený postup implementovat. Knihovna byla naimplementována v programovacím jazyku C# pod platformou .NET. Při návrhu predikce byl kladen důraz zejména na čas a místo, kde se akce uskutečnila, a zároveň na určitou návaznost akcí. Byly tedy uvažovány tzv. sekvence akcí splňující určitá kritéria jako například návaznost na předešlou akci do maximálního časového intervalu. Tyto sekvence mohou uživateli velmi pomoci při vykonávání často se opakující množiny akcí. Vytvořená knihovna je zamýšlena jako součást většího systému zabývajícího se možnostmi pro řízení domácího prostředí vyvíjeného v rámci skupiny NIT (Nature Inspired Technology), do kterého by měla být zakomponována. Celý systém a tedy i knihovna jsou zaměřeny zejména na osoby s omezenou možností pohybu a reakce.

Pro ověření činnosti a testování spolehlivosti predikčního modulu byly náhodně vygenerovány soubory obsahující seznamy akcí fiktivních uživatelů avšak do velké míry simulující chování skutečného uživatele v domácnosti, tzv. logy. Z těchto souborů knihovna vždy vychází a na jejich základě je schopna predikovat akce nejvíce vyhovující aktuálnímu času a místu uživatele. Uvedené souboru rovněž průběžně aktualizuje přidáváním nově vykonaných akcí pro pozdější zpřesňování a adaptaci predikčního modulu.

Přesnost predikce samozřejmě záleží z velké části na konkrétním uživateli. Vykonává-li uživatel často a v přibližně stejnou dobu opakující se akce, predikce je velmi přesná. S rostoucím obsahem základního logu takovéhoto uživatele roste rovněž i úspěšnost predikovaných akcí. Naopak, pokud uživatel nekoná akce dostatečně pravidelně, nebo v přibližně stejných časových okamžicích, tak predikovat příští akci, natož sekvenci, je velmi obtížné a úspěšnost návrhu klesá. U takovéhoto uživatele nedojde ke zvýšení úspěšnosti predikce ani se zvětšením množiny dnů zaznamenaných v logu.

V návrhu je však prostor pro další vylepšení. Například možnost zahrnout do predikce souvislosti mezi akcemi více uživatelů v domácnosti a jejich vzájemné

interakce. Bylo by také možné rozlišovat jednotlivé dny v týdnu, nebo rozdělit dny na pracovní a víkend.

9. Použitá literatura

[1] http://www.cmsys.cz/cze/automatizace-budov/system-domintell/ (stránky společnosti zabývající se automatizací budov)

[2] http://www.itdum.cz/ (internetový magazín o inteligentních domech)

[3] http://www.automatedlifestylesllc.net/ (společnost zabývající se automatizací budov)

[4] Řehoř, M. (2010). Predikce v prostředí inteligentního domova. *Diplomová práce – ČVUT, FEL*.

[5] http://msdn.microsoft.com/en-us/library (Online dokumentace Microsoft .NET Framework, jazyk C#)

[6] http://www.codeproject.com/ (C# utility a příklady)

[7] http://www.automatizace.cz/ (odborný časopis pro automatizaci, měření a inženýrskou techniku)

[8] http://wikipedia.org/ (Wikipedie - otevřená encyklopedie, anglická i česká verze)

10. Přílohy

10.1. Příloha A - obsah doprovodného CD

- BAP_Cerny_Vaclav_Predikce_Akci_2011_05_27.pdf
  Text bakalářské práce

- Actions Prediction
  Složka obsahující zdrojové kódy k softwarové knihovně

- LogGenerator
  Složka obsahující zdrojové kódy k programu náhodně generujícímu log

- Log
  Složka obsahující 13 různých pseudonáhodně vygenerovaných logů.