VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

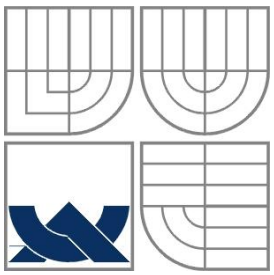
INTELIGENTNÍ SPRÁVA VYZVÁNĚCÍCH PROFILŮ V OS SYMBIAN

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

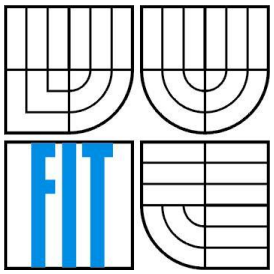
AUTOR PRÁCE
AUTHOR

MAREK HOUŠŤ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTELIGENTNÍ SPRÁVA VYZVÁNĚCÍCH PROFILŮ V OS SYMBIAN

INTELIGENT CONTROL OF RINGING PROFILES IN OS SYMBIAN

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MAREK HOUŠŤ
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ČERMÁK
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Houšť Marek**
Obor: Informační technologie
Téma: **Inteligentní správa vyzváněcích profilů v OS Symbian**
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se operačními systémy v mobilních telefonech, zaměřte se zejména na možnosti přístupu k uživatelkým a provozním informacím.
2. Dle pokynů vedoucího navrhněte aplikaci, která na základě dostupných informací umožní plánovat automatické změny vyzváněcích profilů.
3. Navrženou aplikaci implementujte pro mobilní operační systém Symbian.
4. Zhodnoťte dosažené výsledky, diskutujte případný další možný vývoj projektu a projekt prezentujte malým plakátem.

Literatura:

- Wiley,J. and sons: Programming for the Series 60 Platform and Symbian OS, Chichester, 2003, ISBN 0-470-84948-7
- Windows Mobile Developer Center: http://msdn.microsoft.com/en-us/windowsmobile/
- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Čermák Martin, Ing.**, UIFS FIT VUT
Konzultant: Koutný Jiří, Ing., UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá operačními systémy v mobilních telefonech, se zaměřením na operační systém Symbian. Popisuje návrh, implementaci a testování aplikace pro inteligentní správu vyzváněcích profilů. Aplikace je implementována v operačním systému Symbian. Na závěr je diskuze o budoucím rozvoji aplikace.

Klíčová slova

Mobilní zařízení, mobilní telefon, smartphone, Palm, Android, LiMo Foundation, RIM BlackBerry, iPhone OS X, Windows mobile, Symbian, vyzváněcí profily, aktivní objekty, Profile Engine Wrapper, Symbian signed

Abstract

This bachelor thesis is about operations systems in mobile phones focusing on operation system Symbian. It describes proposal, implementation and testing of application for intelligent control of ringing profiles. Application is implemented in operation system Symbian. There is discussion about future development in the end.

Keywords

Mobile device, mobile phone, smartphone, Palm, Android, LiMo Foundation, RIM BlackBerry, iPhone OS X, Windows mobile, Symbian, ringing profiles, active object, Profile Engine Wrapper, Symbian signed

Citace

Houšť Marek: Inteligentní správa vyzváněcích profilů v OS Symbian. Brno, 2009, bakalářská práce, FIT VUT v Brně.

Inteligentní správa vyzváněcích profilů v OS Symbian

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Čermáka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Marek Houšť
20. 5. 09

Poděkování

Děkuji panu Ing. Martinovi Čermákovi za vedení bakalářské práce a panu Ing. Jiřímu Koutnému za připomínky a doporučení.

Obsah

Úvod

Hlavním cílem této práce je naprogramování aplikace, která bude inteligentně spravovat vyzváněcí profily v operačním systému Symbian. Jelikož tento OS je od jeho počátku navržen a používán výhradně v mobilních zařízeních, tak *první kapitola* práce tyto mobilní zařízení popisuje. Tato kapitola také definuje oblast mobilních telefonů typu Smartphone, kde je tento OS Symbian nejpoužívanějším systémem na světě (podrobněji zdůvodním v *kapitole 2.1*). Existují ale i další operační systémy, které jsou v mobilních zařízeních používány.

Druhá kapitola seznamuje s nejčastěji používanými operačními systémy v mobilních telefonech včetně OS Symbian. Tato kapitola informuje o jejich možnostech přístupu k uživatelským informacím. Objasňuje především to, jak systém podporuje vývoj aplikací vývojáři třetích stran.

Třetí kapitola definuje návrh aplikace pro inteligentní správu vyzváněcích profilů pod OS Symbian. Kapitola popisuje to, jaké funkce by taková aplikace měla mít s ohledem na prostudování technologií a operačního systému jako celku. Jelikož mobilní telefony s OS Symbian podporují pouze manuální přepínání uložených vyzváněcích profilů, tak bude program pro inteligentní správu vyzváněcích profilů velmi atraktivní pro uživatele. Tyto uložené vyzváněcí profily lze v OS Symbian editovat a přidávat nové. Aplikace by proto měla umět pracovat s aktuálními a hlavně všemi uloženými vyzváněcími profily. Aplikace pro správu vyzváněcích profilů od vývojářů třetích stran již samozřejmě existují. Příkladem může být program *Handy Profiles* od společnosti Epocware [1].

Kapitola čtvrtá popisuje implementaci navržené aplikace z *kapitoly třetí*. Zpočátku se zabývá výběrem platformy, vývojového nástroje a programovacího jazyka. Dále seznamuje se zásady programování pod OS Symbian. Poté jednotlivé kapitoly popisují použité technologie OS Symbian nutné pro funkčnost aplikace. Také popisuje důvody jejich použití a způsob jejich implementace. Závěr kapitoly se věnuje bezpečnostním opatřením přístupu ke zdrojům OS Symbian.

Pátá kapitola obsahuje testování implementované aplikace z *kapitoly čtvrté*. Popisuje průběh a příklady testů. Tyto testy na sebe navazují a jsou uživatelskými scénáři, které by v reálném provozu mohly nastat.

Na *závěr* se diskutují celkové dosažené výsledky, použitelnost aplikace uživatelem v reálném provozu a případný další rozvoj aplikace do budoucna.

1 Mobilní zařízení

Kapitola popisuje mobilní zařízení, pod kterými si lze představit multi-funkční zařízení schopné téměř suplovat práci na PC s tím, že práce mimo kancelář či domov lze pak synchronizovat s vlastním PC. Většinou se jedná o zařízení, která spolu s funkcí mobilního telefonu jsou schopná přistupovat k internetu a emailu. Mezi samozřejmé funkce dnes již patří také digitální fotoaparát, MP3 přehrávač a další. Za mobilní zařízení lze dnes označit mnoho elektroniky.

1.1 Tablet PC

Za mobilní zařízení lze považovat i klasické notebooky. Tablet PC mají nejméně rozdílů mezi kapesním počítačem a notebookem, přičemž jsou více uzpůsobeny na mobilitu. Mají klasickou klávesnici a vstup pro myš. Tablet PC používají většinou operační systém Microsoft Windows XP Tablet PC Edition. Jedná se o klasický OS Windows XP, který je doplněn o ovladače pro podporu tablet hardwaru. OS Windows Vista již nepotřebuje zvláštní edici pro podporu tohoto hardwaru. OS Windows Vista podporuje i v dnešní době poměrně moderní MultiTouch tablet. Tyto tablety jsou schopny snímat více dotyků najednou. V dnešní době podporu pro tablet PC podporují i Mac OS X a některé verze Linuxu. Jedná se také o klasický systém, který je doplněn o podporu používání pera (stylus).

Obrázek 1: Nova Mobility Solution, SideArm2 (převzato z [2])

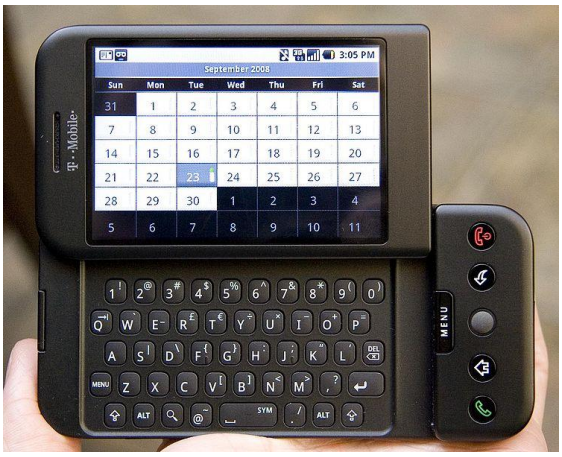
1.2 Personal Digital Assistant (PDA)

PDA se již za kapesní zařízení dají považovat. Původně měly pouze pomáhat s organizací času a kontaktů, jak již naznačuje překlad slov Personal Digital Assistant. Dnes fungují i jako mobilní telefony s širokými funkcemi. Nemají klávesnici, jaká je u klasického PC, ale obsahují dotykovou obrazovku a pero. Text se zadává pomocí systémů rozpoznávajících písmo. Základem je

synchronizace se stolním počítačem. Typickými operačními systémy u PDA jsou Windows Mobile, Palm OS a Symbian OS. Popis těchto systémů se nachází v *kapitole 2*.

1.3 Smartphone

Telefony typu Smartphone se vyznačují kombinací mobilních telefonů a kapesních počítačů (PDA). Většinou obsahují nějaký otevřený, proprietární, či uzavřený operační systém (Symbian OS, Windows OS, OS X, Android, PalmOS). Přístroje předpokládají budoucí rozšíření o další aplikace, jejich operační systémy jsou na to uzpůsobeny. Vydávají vývojářské kity (tzv. SDK) zahrnující většinou tutoriály, ukázkové programy, dokumentace, knihovny pro určitá aplikační programová rozhraní, nebo i přímo vývojová prostředí, emulátory a debuggery. Existují také fóra, na kterých se řeší problémy vývojářů třetích stran.



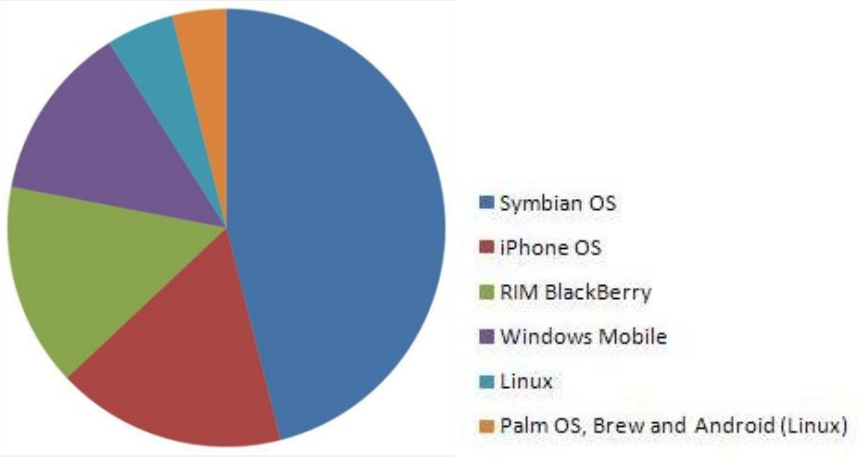
Obrázek 2: HTC, T-mobile G1 s OS Android (převzato z [3])

2 OS v mobilních telefonech

V kapitole je základní popis operačních systémů v mobilních telefonech, se zaměřením na možnosti přístupu k uživatelským a provozním informacím. Zaměřuje se především na to, jak jednotlivé systémy umožňují vývoj aplikací vývojáři třetích stran. *Úvod* popisuje důvod výběru jednotlivých operačních systémů popisovaných v podkapitolách dále. Kapitola obsahuje *Palm OS*, *OS mobile založené na operačním systému Linux*, *RIM BlackBerry OS*, *iPhone OS X*, *Windows Mobile* a *Symbian OS*. Posledním jmenovaným se zabývá více, jelikož pod tímto OS bude navržena a implementována aplikace pro inteligentní správu profilů.

2.1 Úvod

Operační systém v mobilních telefonech je spojen s konkrétním mobilním přístrojem, je zodpovědný za stanovení funkcí na konkrétním mobilním telefonu. Také určuje, jaké aplikace od třetích stran smějí být instalovány na daný systém.



Obrázek 3: Podíl na trhu ve 3. čtvrtletí 2008 (převzato z [4])

Nejprve bylo nutné vybrat, na které operační systémy se zaměřit. Jelikož vývoj mobilních telefonů je jedním z nejrychleji se vyvíjejících odvětví na světě, tak jsem se nakonec rozhodl vybrat jednotlivé OS podle posledních statistik prodejů mobilních telefonů na trhu. Na grafu lze vidět, že mobilní telefony s OS Symbian se stali nejprodávanějším na světě i přes nástup novinky iPhone OS.

2.2 Palm OS

Je proprietární operační systém, navržen pro mobilní zařízení PDA. Jeho počátek je ve firmě Palm Computing, kde byl vyvinut pod vedením Jeffa Hawkingse. Od verze Paml OS 5.4 je známý také jako Garnet OS, či od verze 6.0 jako Palm OS Cobalt. Dříve byl OS single-tasking, což znamená, že při

přepínání mezi aplikacemi si aplikace uložila svůj poslední stav. Později operační systém umožňoval spustit několik aplikací na pozadí. Až verze Palm OS Cobalt (2004) zavedla moderní OS na novém jádře s podporou multitaskingu, ochranou paměti a moderními grafickými rámcem. Nyní se očekává nástup nové verze Paml OS založené na operačním systému Linux s názvem Nova (2009). [5]

Pro vývojáře třetích stran existují licence pro vlastní úpravy operačního systému, knihovny API pro telekomunikaci, Wifi, Bluetooth, digitální kameru, knihovnu pro podporu vstupů a další. Ty se nacházejí ve vývojovém balíčku SDK, který lze získat po zaregistrování na webové stránce Palm Developer Network. Aplikace jsou převážně psány v C/C++. Existuje kompilátor CodeWarrior Development Studio for Palm, Palm OS Developer Suite a open-source nástroje PRC-tools, které jsou založené na staré verzi kompilátoru GCC. Existuje i kompilátor OnBoardC, který běží přímo na přístroji Palm. [5, 6]

2.3 OS mobile založené na systému Linux

Projekty založené na systému Linux se většinou snaží tvořit mobilní telefony s otevřenou softwarovou platformou. Např. Projekt Openmoko se snaží naplňovat hesla Open, Mobile, Free. Na daných telefonech poběží jakýkoliv systém založený na Linuxu. [7]

Mezi OS mobile distribuce založené na systému Linux patří např.: FDOM, Qtopia, FSO, Android, LiMo Foundation Platform.

2.3.1 Android

Tato softwarová platforma a operační systém převážně pro mobilní zařízení je založena na linuxovém jádru verze 2.6, které působí jako abstraktní vrstva mezi hardwarem a částí softwarovou. Tento OS byl vyvinut společností Google (2008). Ta také vyvinula knihovny SDK, které umožňují vývojářům psát aplikace pro tento systém v jazyce Java. Tento Software Development Kit obsahuje kompletní sadu nástrojů pro tvorbu. Rozvoj je podporován pro platformy Linux, Mac OS X, Windows XP, či Vista. Oficiální vývojové prostředí je Eclipse. Systém je k dispozici jako open-source a je navržen tak, aby mohl běžet na téměř veškerém hardwaru. Systém je kompletně multitasking. [8, 9]

2.3.2 LiMo Foundation Platform

Tato LiMo (**Li**nux **Mo**bile) nadace byla založena společnostmi Motorola, NEC, NTT DoCoMo, Panasonic Mobile Communications, Samsung Electronics a Vodafone pro spolufinancování rozvoje mobilní platformy založené na Linuxu (2007). Platforma LiMo Foundation je vyvíjena s cílem umožnit vývoj mobilních telefonů, které budou založeny na modulární, plug-in architektuře, která bude postavena na otevřeném operačním systému. Využívá již existujících standardů a dalších open-source projektů. Aplikace mohou přistupovat ke komponentám prostřednictvím definovaných

rozhraní API. Komponenty Application Manager Application Framework a UI Framework mají rozhraní v jazycích C/C++. Platforma je navržena tak, aby byla nezávislá na hardwaru. [10, 11]

2.4 RIM BlackBerry OS

Je proprietární multitasking systém pro mobilní telefony BlackBerry. Systém je vyvíjen kanadskou společností Research In Motion. Vývojáři třetích stran mohou psát sotfware pomocí rozhraní knihoven API. Existují otevřené knihovny pro vývoj, ty ale dodávají aplikacím omezenou funkčnost. Větší funkčnost dodávají knihovny tzv. Signed API, ty však musí být digitálně podepsané. Mezi vývojářem a společností RIM pak existuje účet. Tento účet poté jednoznačně identifikuje vývojáře. Vývoj je podporován v jazyce Java. Integrované vývojové a simulační prostředí tzv. Java Development Environment je nainstalováno jako plug-in do vývojového prostředí Eclipse. Toto prostředí obsahuje včetně knihoven API také ukázkové aplikace. [12]

2.5 iPhone OS X

Tento OS je vyvíjen společností Apple Inc. pro iPhone a iPod Touch. Sdílí hodně vlastností a technologií s Mac OS X. Nelze ale aplikace z Mac OS X jednoduše zkopírovat a spustit na iPhone OS. Existuje iPhone SDK, který obsahuje ukázkové kódy, informace a nástroje pro vývoj a testování aplikací. Pro vývoj však potřebujeme počítač s Mac OS X a nástroj Xcode. Po vývoji v simulačních nástrojích se musí zaplatit poplatek za možnost nahrát program do přístroje.

Architektura iPhoneOS se skládá ze 4 vrstev, a tomu je přizpůsobeno i rozdělení vývojového kitu SDK. Nejspodnější vrstva se jmenuje Core OS, ta je založena na stejném jádru Mach jako Mac OS X. Tato nejspodnější vrstva například spravuje virtuální paměťový prostor, souborový systém, síťové služby (BSD sockets) a vlákna. Nad jádrem běží vrstva Core Services, pomocí které se přistupuje k vlastnostem základní vrstvy. Vyšší vrstvy se jmenují Media a Cocoa Touch. Vrstva Media spravuje přístup k multimediálním službám. Poslední vrstva Cocoa Touch obsahuje grafické prvky systému a uživatelské informace. [13]

2.6 Windows Mobile

WM je uzavřený operační systém. Uživatelské rozhraní zůstává nezměněno v celém zařízení. Windows Mobile SDK poskytuje vývojářům nástroje a rozhraní API pro vývoj aplikací ve Visual Studiu. SDK jsou i přes uzavřený operační systém k dispozici zdarma, verze závisí na platformě, pro kterou se aplikace vyvíjí. SDK také zahrnuje celou řadu emulátorů pro zařízení Windows Mobile, které jsou nainstalovány společně s WM SDK a jejich komponenty jsou zařazeny do Visual Studia.

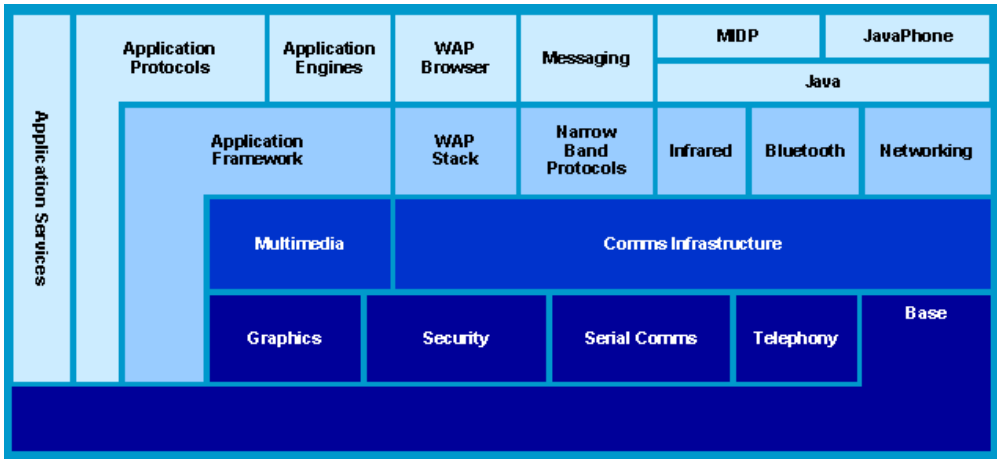
Rozhraní API jsou uspořádány do různých sekcí. **Applications and Services**, kde můžeme najít nástroje pro synchronizaci dat s OS Windows, ovladač pro GPS, API pro psaní aplikací užívající adresáře LDAP a subsystém pro manipulaci s emailovými a textovými zprávami (SMS). **Audio**, kde lze najít zvukový subsystém, který spravuje kodeky, konverzi audio formátu a audio filtry. **Encoded Media** obsahuje technologie pro zakódovaná multimediální data. **File Systems and Storage Management** obsahuje rozhraní pro několik typů souborových systémů. **Graphics Device Interface (GDI)** poskytuje grafickou podporu pro využití speciálního grafického hardware. **Networking** obsahuje informace o obecné technologii podporovaných sítí. V této sekci lze nalézt témata jako TCP/IP, Windows Sockets, Bluetooth, Wi-fi.
[14, 15]

2.7 Symbian OS

Je to proprietární operační systém, který byl navržen firmou Symbian Ltd, která byla založena roku 1998 firmami Ericsson, Nokia, Motorola a Psion. Dnes (2009) je vlastněna pouze společností Nokia a systém je spravován nadací Symbian. Systém není open source, avšak knihovny API jsou zdokumentovány a veřejně přístupné.

Pro vývoj OS Symbian je důležitý Symbian Developer Network, který poskytuje technické dokumenty a tutoriály, vývojářské kity SDK, fórum pro vývojáře, wiki pro OS Symbian a Symbian Developer knihovnu, která popisuje jednotlivé verze OS Symbian. Dále pak obsahuje dokumentaci systému, ukázkové kódy a často kladené otázky. Důležité jsou především nástroje pro vývoj a vývojářské kity SDK. Výborným nástrojem pro programování aplikací je nástroj **Carbide.c++**. Tento nástroj je integrované vývojové prostředí, založené na open source platformě Eclipse. Toto IDE existuje ve třech verzích - profesional, developer a OEM. Dále existuje GCC překladač pro Symbian. Pro psaní aplikací vývojář potřebuje vývojový kit SDK pro určitou platformu. Je tedy nutné při vývoji pro konkrétní telefon nejdříve zjistit jakou verzi OS Symbian telefon používá. SDK podporuje vývoj v jazycích C++, Java a Python. Obsahuje kompletní dokumentaci systému a příklady již napsaných aplikací, ke kterým existují zvlášť dokumentace. [16, 17]

OS Symbian se stává z několika vrstev, které nazýváme subsystémy.



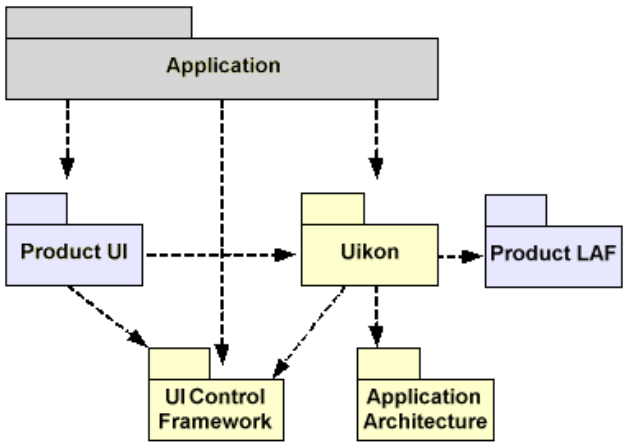
Obrázek 4: Jednotlivé subsystémy OS Symbian (převzato z [18])

Základní subsystém **Base** se stará o manipulaci s deskriptory, pomocí kterých se v OS Symbian zachází s řetězci. Je to prakticky objekt, který se nachází na určitém místě v paměti. Objekt obsahuje data, ukazatel na ně a délku dat. Dále se stará například o preemptivní multitasking pomocí vláken či nepreemptivní multitasking pomocí jednoho vlákna a aktivních objektů. Základní subsystém se stará také o napájení, ovladače a souborový systém. Soubory jsou identifikovány pomocí specifikace, která může mít až 256 znaků. Systém podporuje 26 jednotek. Znak `z:` je většinou systémová paměť ROM, `c:` interní paměť pro čtení a zápis a znak `d:` je většinou vyměnitelné médium. [17, 19]

Subsystém zabezpečení **Security** obsahuje rozhraní pro kryptografické algoritmy, možnosti generování hashovacího klíče, náhodných čísel, správu certifikátů a instalační program. Subsystém grafiky **Graphics** poskytuje rozhraní pro kresbu, vkládání obrázků, písma a animací. Poskytuje také tisk na různých tiskárnách pomocí grafického kontextu. Obsahuje také rozhraní pro přidání ovladače tiskárny. Subsystém **Telephony** je určen pro přístup k funkcím mobilního telefonu, také obsahuje pomyslnou kostru pro psaní ovladačů pro hardware telefonu. Subsystém, který definuje základní strukturu aplikace a základní uživatelské rozhraní Uikon se nazývá **Application Framework.** Výrobci si sem přidávají své vlastní komponenty nutné k provozu jejich zařízení. Subsystém **Application Engines** dodává přístup do základních aplikací jako je správa kontaktů a organizační data ve formě kalendáře. Nové aplikace tak můžou mít přístup například k čtení, ukládání a mazání položek v kalendáři, či kontaktech. Pomocí subsystému **Application services** přistupujeme k systémovým informacím včetně použití Alarm serveru a užívání data a času. Subsystém **Application protocols** zahrnuje komponenty pro přístup k webu pomocí http protokolu, včetně nástrojů jako parsování a validace URI. Nástroj pro komunikaci pomocí sockets můžeme najít v **Comms Infrastructure**, ty jsou podobné BSD sockets. Na tento subsystém poté navazují subsystémy jako **Infrared**, **Bluetooth** a **Networking.** [17, 19]

2.7.1 Uživatelská rozhraní

OS Symbian uživatelské rozhraní odděluje od operačního systému. To je tvořeno knihovnami z vrstvy systému zvané UI Framework, která patří do subsystému Application Framework. Nachází se nad jádrem operačního systému. Základem je jádro Uikon, to již definuje množství ovládacích prvků a dialogů. V dnešní době existují různá uživatelská rozhraní, která jsou určena pro různé druhy mobilních telefonů. Tato různá uživatelská rozhraní mají společné jádro Uikon a běží na stejném jádru operačního systému. Protože se telefon od telefonu liší požadavky na uživatelské rozhraní, tuto odlišnost do uživatelského rozhraní doplňují až výrobci telefonu.



Obrázek 5: Architektura uživatelského rozhraní OS Symbian (převzato z [18])

Na *obrázku 5* můžeme vidět jádro Uikon. Knihovny, které dodal výrobce telefonu, jsou obsaženy v Product UI a Product LAF, kde najdeme například velikost ovládacích prvků jádra Uikon a barvy. UI Control Framework a Application Architecture se starají o inicializaci aplikace při jejím startu, o detekci vstupu od uživatele a jeho zpracování, a také o interakci se souborovým systémem. Existuje několik odlišných platforem uživatelského rozhraní založené na Symbian OS. Jsou to series 60, series 80, UIQ. Nejrozšířenější, a v dnešní době nejvíce podporovaná ve vývoji, je platforma series 60. U platformy series 60 se nadstavba nad základním jádrem Uikon jmenuje Avkon. Podle počátečních písmen Akn lze také poznat, zda je nejrůznější prvek součástí této platformy. [19]

3 Návrh aplikace

Tato kapitola popisuje návrh aplikace pro inteligentní správu vyzváněcích profilů v OS Symbian. Návrh aplikace byl vytvořen na základě dostupných znalostí o tomto systému. Prvně bylo nutné si udělat přehled o ovládacích prvcích, které se při vývoji aplikací s uživatelským rozhraním používají. Jelikož je nejrozšířenější platformou OS Symbian platforma series 60, vybral jsem si k prostudování tyto ovládací prvky. Tyto ovládací prvky jsou pro mě poměrně odlišné od doposud známých prvků z operačních systémů Windows či Linux. Při návrhu funkčnosti byl kladen důraz na to, aby práce s programem uživateli zabrala co nejméně času. Podle toho byly navrženy i informace, na základě kterých bude umožněno plánování automatických změn profilů. Pomocí tohoto návrhu byla poté aplikace implementována (*kapitola 4.*) a testována (*kapitola 5.*). Návod k implementované aplikaci včetně ukázkových obrázků lze najít v *příloze 1*.

3.1 Základní funkce

Aplikace, která inteligentně spravuje vyzváněcí profily, by měla být schopna měnit za daných informací definované profily v operačním systému. Tyto informace budou zadávané uživatelem a budou popsány později. Aplikace také bude načítat a pracovat dynamicky s aktuálními profily[1]. Při případném vymazání profilu, který bude figurovat v nějakém nastavení, bude nahrazen profilem Normální (General). Aplikace se bude spouštět ihned po startu systému. Bude ji možné tlačítkem z panelu ovládacích prvků přesunout na pozadí. Při jejím přesunutím na popředí se bude zobrazovat přehled definovaných informací. Bude k tomu sloužit prvek z uživatelského rozhraní Avkon, List Box.

Prvek List Box bude umožňovat označení více informací najednou. Toto bude užitečné při práci s více uloženými informacemi současně (editace, mazání). Uživatel bude moci označit či odznačit jednu či více podmínek (o podmínkách více v *kapitole 3.2.1*). Také bude moci označovat podmínky logicky podle jejich nastavení:

- hlavním tlačítkem joysticku (`[ok]`) bude probíhat označování či odznačování jednotlivých uložených podmínek,
- pomocí menu nebo klávesových zkratek (`[7]` a `[9]`) bude možné označit či odznačit všechny podmínky z jedné množiny,
- pomocí menu nebo klávesových zkratek (`[8]` a `[0]`) bude možné označit či odznačit všechny podmínky ze stejné množiny se startovními daty ve stejný den v týdnu,

[1] V OS Symbian je běžné, že lze přidávat libovolné množství profilů a stávající přejmenovávat.

- pomocí menu nebo klávesových zkratek (`[*]` a `[#]`) bude možné označit či odznačit všechny uložené podmínky.

Aplikace bude mít i na další její funkce (budou popsány dále) klávesové zkratky, které budou zobrazeny u jednotlivých funkcí v menu.

3.2 Definované informace

Automatické změny profilů budou probíhat na základě dvou možných nastavení. Uživatel bude moci nastavovat neomezené množství podmínek, které se budou při ukončení aplikace ukládat, a budou pro události známé dopředu. Toto množství podmínek bude omezeno pouze pamětí telefonu. Pro události, které se vyskytnou náhle, bude sloužit nastavení změna na čas.

3.2.1 Podmínky

Toto nastavení bude uživateli umožňovat velmi rychlou správu všech změn vyzváněcích profilů na libovolnou dobu dopředu. Například provede vygenerování podmínek, které se postarají o změnu profilů po určité období, den v týdnu a čas. Poté pokud nastane změna a profily bude nutné měnit v jiný čas či datum, tak se podmínky pomocí hromadného označování rychle hromadně zeditují. Výhodou je, že se nemusí podmínky na určité období nastavovat a editovat pro každý den zvlášť, i když toto nastavení aplikace bude také podporovat.

Podmínky tedy bude možné generovat po množinách, nebo přidávat samostatně. Poté je bude možné editovat a mazat hromadně či samostatně. Každá podmínka bude obsahovat parametry:

- aktivitu – udává, jestli se bude podmínka ve změnách profilů projevovat,
- název,
- profil – udává, kterého profilu se daná podmínka týká,
- konkrétní čas a datum změny na nastavený profil,
- konkrétní čas a datum změny zpět na profil před změnou.

Tyto parametry podmínek budou vygenerovány pomocí ovládacích prvků ze seznamu Setting Item List, který samozřejmě také patří do uživatelského rozhraní Avkon. Profil, na který se bude měnit zpět (profil před změnou) bude ke každé podmínce doplněn až za běhu. Pokud bude čas a datum změny a změny zpět shodný, tak ke změně zpět na profil nedojde. Jinak bude čas a datum změny a změny zpět označovat rozmezí, ve kterém se na profil změní. Například pokud uživatel zapne program, až když bude aktuální čas a datum později než čas a datum změny, ale dříve než čas a datum změny zpět, tak se na profil také změní.

Při generování podmínek stejné množiny bude nutné tedy nastavovat:

- název množiny podmínek (bude použit jako název všech podmínek z množiny),
- profil – udává, na jaký profil se budou podmínky z množiny měnit,

- čas změny – udává pouze čas, v kterém budou podmínky začínat,
- čas změny zpět – udává pouze čas, v kterém budou podmínky končit,
- datum startu období generování,
- datum konce období generování,
- po kolika dnech v nastaveném období se budou podmínky generovat.

Aplikace pomocí tohoto nastavení po potvrzení a kontrole zadaných údajů začne generovat podmínky v daném rozmezí dat. Pokud bude čas změny zpět menší než čas změny tak podmínka bude končit následující den.

Při přidávání jedné podmínky se nebude zadávat rozmezí generování. Pouze se bude nastavovat společně s ostatními hodnotami den startu podmínky. Při generování a přidávání se bude parametr aktivity automaticky považovat za aktivní.

Při hromadné editaci podmínek bude možné editovat:

- aktivitu,
- profil,
- čas změny,
- čas změny zpět.

Datum startu bude možný editovat pouze při editaci jedné podmínky. Hromadně se budou moci editovat pouze ty parametry podmínek, které budou stejné. To znamená, že půjdou hromadně editovat pouze podmínky ze stejné množiny (stejného názvu). Toto omezení je z důvodu větší přehlednosti. Dané parametry se při editaci budou načítat zpět do ovládacích prvků seznamu Setting Item List. Tak uživatel přehledně uvidí, co edituje. Pokud uživatel například označí podmínky, které budou mít stejný pouze parametr profil, ostatní parametry budou skryty. Dále nelze editovat či danou editaci uložit pokud podmínka již bude probíhat. Taková podmínka půjde pouze smazat, i tak na to uživatel bude upozorněn varovným hlášením. Po smazání takové podmínky se ihned provede přechod na původní profil.

Při ukládání či znovu ukládání (editaci) určitých podmínek bude muset být kontrolováno:

- datum startu a konce generování (zda není startovní datum zadán později jako koncový),
- datum startu (zda jej uživatel nezadal tak nízký, že by po potvrzení údajů a vygenerování podmínek se v zápětí smazaly díky tomu, že program staré podmínky po týdnu maže),
- překryv podmínek (zda nějaká ukládaná podmínka nepřekrývá podmínku již uloženou).

Tato kontrola je oznamována uživateli varovným hlášením. Kontrola překryvu podmínek se pro uživatele ukázalo být nejpoužitelnější. Toto je odlišnost od původního návrhu. Nejprve jsem se snažil navrhnout a později implementovat systém, který by inteligentně spravoval podmínky s možností překrývání se. Při překrývajících se podmínkách by se koncové změny zpět uskutečňovaly pouze u podmínek, do kterých by nevstupovala jiná. *Obrázek 6* toto názorně demonstruje.

1. start 1. podmínky:	Mění se na profil z **1.** podmínky.
2. start 2. podmínky:	Mění se na profil z **2.** podmínky.
3. start 3. podmínky:	Mění se na profil z **3.** podmínky.
4. konec 3. podmínky:	Ukončí se, mění se zpět na profil z **2.** podmínky
5. start 4. podmínky:	Mění se na profil ze **4.** podmínky
6. konec 1. podmínky:	Ukončí se, nemění se zpět (běží 2. a 4. podmínka)
7. konec 4. podmínky:	Ukončí se, mění se zpět na profil z **2.** podmínky
8. konec 2. podmínky:	Ukončí se, mění se zpět na profil, který byl před všemi změnami (na profil před 1. podmínkou).

Obrázek 6: Překrývající se podmínky

Toto se ale později ukázalo za velice nevhodné řešení, které by se uživateli mohlo jevit až jako náhodné. K tomuto názoru jsem došel s ohledem na to, že uživatel by si nemusel uvědomit, že daná podmínka vůbec jinou překrývá. Uživatel totiž podmínku, která by překrývala jinou, mohl zadat například až za měsíc. Muselo proto dojít k přepracování návrhu a částečně i implementace.

Podmínky se nebudou po vypršení mazat ihned. Aplikace bude podmínky, které proběhly, v paměti ponechávat týden staré, aby bylo možné se k nim případně vracet. V aplikaci bude nutné také ošetřit to, pokud se uživatel pokusí vygenerovat více podmínek, než na které bude stačit paměť.

3.2.2 Změna na čas

Nastavení změny na čas je vhodné, když uživatel chce jednu či více postupných změn profilů ihned bez ohledu na definované podmínky. Uživatel nastaví jednoduše, který profil jak dlouho bude trvat. Na první nastavený profil se přejde ihned, tento pak bude trvat po nastavenou dobu. Pokud uživatel těchto změn nastavil více, tak se po této změně přejde na druhý, který bude trvat opět zadanou dobu. Po ukončení všech nastavených změn se přejde na profil, který byl před všemi změnami.

Toto natavení bude moci uživatel pouze nastavit či zrušit. I při zrušení se provede přechod na původní profil ihned. Aktivita či neaktivita změny na čas se bude zobrazovat také v seznamu List Box hned na prvním místě. Při aktivní změně na čas bude také viditelná neaktivita uložených podmínek. Parametr aktivity podmínek také nebude moci být editován při aktivní změně na čas. Pokud bude uživatel chtít změnu na čas spustit a bude nějaká podmínka probíhat, tak na to bude upozorněn. Pokud při ukončení změny na čas, bude aktuální čas v rozmezí startovního a koncového času nějaké podmínky, tak se přejde na profil podle dané podmínky.

4 Implementace aplikace

Kapitola popisuje implementaci navržené aplikace z *kapitoly 3* pro OS Symbian. Popisuje výběr a důvody použité platformy a vývojových nástrojů. Věnuje se zásadám programování pro OS Symbian. Dále popisuje základní třídy a knihovny aplikačního rozhraní, které jsou nutné pro vývoj aplikace s grafickým rozhraním. Jelikož jsou pomocí tohoto rozhraní informace potřebné pro inteligentní správu vyzváněcích profilů ukládány, tak se věnuje také způsobu ukládání informací a jejich vyhledáváním. Popisuje také použité technologie nutné pro funkčnost aplikace. Návod k implementované aplikaci, včetně všech funkcí aplikace a názorných obrázků lze najít v *příloze č. 1*. Proto se v této kapitole zabývám pouze implementací.

4.1 Výběr platformy a vývojové nástroje

Na počátku je třeba si vybrat, podle telefonu, pro jakou platformu budeme chtít aplikaci vyvíjet. Jelikož již v návrhu byla zmíněna platforma series 60, tak jsem se rozhodl pro platformu **S60 3rd Edition FP1 (Symbian OS v9.2)**. Tuto verzi systému můžeme nalézt v telefonech Nokia N95, N82, N77, Samsung SGH-i450, SGH-i550, SGH-i520, SGH-i560, G81 a LG JOY. Je to především z důvodu vlastnictví mobilního telefonu s touto platformou.

Jako vývojové prostředí jsem zvolil Carbide.c++ Developer. Tato verze obsahuje všechny potřebné nástroje po vývoj. Je určená pro nadšence, studenty, ale i menší vývojáře. Implementovat aplikaci budu v programovacím jazyce C++. Je pro OS Symbian nejpoužívanější a umožňuje využít plný potenciál platformy.

4.2 Pojmenování tříd

Pro vývoj aplikací pro OS Symbian je důležité používat určité zásady pojmenování pro jednodušší pochopení zdrojového kódu. Vždy se jedná o počáteční písmeno jak u tříd, tak u jmen dat.

- **`T:`** Třídy datových typů, které se chovají jako typy vestavěné.
- **`C:`** Třídy, které jsou vždy alokovány dynamicky na haldě. Při alokaci se všechna členská data inicializují na nulu.
- **`R:`** Třídy pro přístup ke zdrojům operačního systému (například časovač `RTimer`). Většina používá pro uvolnění zdrojů (dealokaci) funkci `Close()`.
- **`M:`** Třídy, které definují rozhraní. Pouze ony můžou použít vícenásobnou dědičnost.
- **`S:`** Struktury, které jsou podobné klasickým strukturám ze standardního jazyka C.
- **`E:`** Výčtové konstanty.

- **K:** Konstanty, které jsou definovány jako `#define` nebo `const TInt`.
- **i:** Nestatické členské proměnné tříd.
- **a:** Parametry funkcí.

[19, 20]

4.3 Soubory projektů pod OS Symbian

- **rss/rssi:** Jsou tzv. resources, které slouží k definování různých pohledů, dialogů, menu případně definují informace o aplikačním jméně a identifikačním čísle `UID` aplikace. Dají se použít také k definování řetězců, pokud je aplikace psána ve více jazykových verzích. Po kompilaci tyto soubory mají příponu `rsc` a jsou binární. Pokud jsou kompilovány pro určitou jazykovou verzi, tak mívají přípony podle určitého jazyka. Například pro český jazyk je to přípona `r25`.
- **h:** Klasické hlavičkové soubory známé ze standardního jazyka C++.
- **hrh:** Tyto soubory jsou většinou použity pro datový typy `enum`.
- **cpp:** Klasické zdrojové soubory známé ze standardního jazyka C++.
- **bld.inf:** Tento soubor používá utilita `bldmake` ke generování skriptu `abld.bat`. Nalezneme v něm také ukazatel na soubor `mmp`, který popisuje projekt.
- **mmp:** Soubor, který popisuje projekt. Definuje zdrojové kódy aplikace a použité knihovny pro `Makefile`, aby bylo možné daný projekt zkompilovat pro cílovou platformu.
- **abld.bat:** Skript, který se stará o generování `Makefile`.
- **Icons_aif_scalable_dc.mk:** Soubor, který definuje vlastní ikonu aplikace typu `sgv`.
- **svg:** Ikona aplikace, která je zobrazená v menu před spuštěním aplikace.
- **bmp:** Ostatní bitmapy aplikace. Každá bitmapa aplikace musí mít dvoubarevnou masku stejných rozměrů (také soubor `bmp`).
- **pkg:** Soubor, který definuje instalační balíček `sis` (popisuje instalaci).
- **sis:** Konkrétní instalační soubor pro konkrétní platformu.

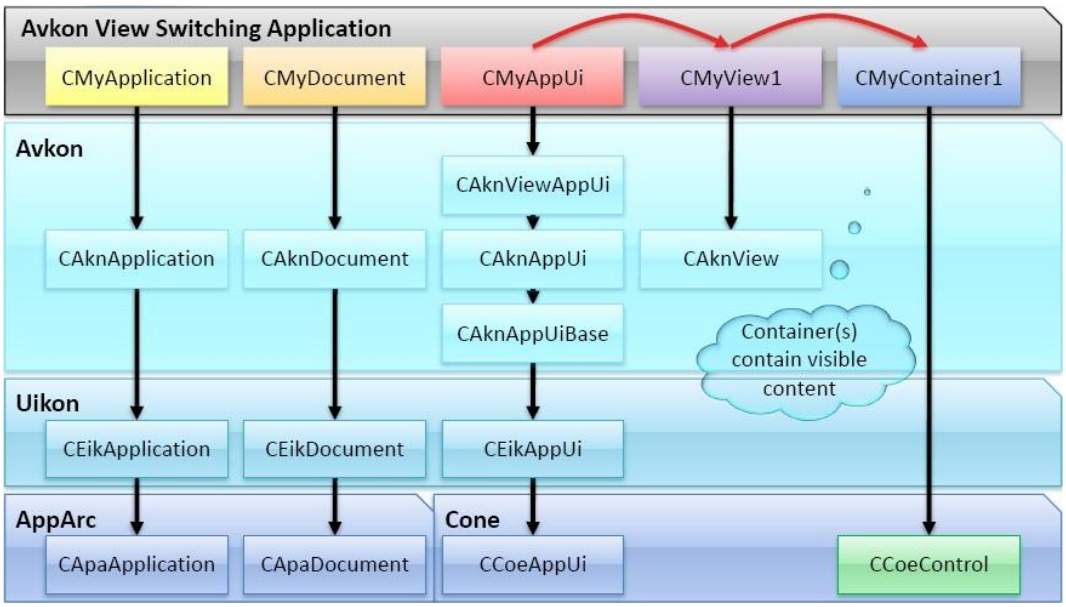
[18] [21]

4.4 Uživatelské rozhraní Avkon

Je vystavěno nad základní vrstvou uživatelského rozhraní `Uikon`. Obsahuje tři ze čtyř základních tříd nutných pro spuštění programu s grafickým uživatelským rozhraním. Jedná se o třídy: `CAknApplication`, `CAknDocument`, `CAknAppUi` a `CCoeControl`. Všechny základní třídy

vrstvy Avkon mají před názvem tři písmena Akn a jsou odvozeny od tříd základní vrstvy Uikon. Třídy základní vrstvy Uikon mají prefix Eik.

OS Symbian jako první volá funkci NewApplication(), jako vstupní bod slouží funkce E32Main(). Funkce NewApplikacion() vytváří a vrací objekt aplikace CPCH_v110Application, třídy odvozené od CAknApplication. Tato funkce zapouzdřuje hlavní aplikační proces a je jakousi obdobou klasické funkce C/C++ main(). Při inicializaci třídy CPCH_v110Application se volá metoda CreateDocumentL(). Tímto se vytvoří třída CPCH_v110Document odvozená od třídy CAknDocument, která je zodpovědná za vytvoření uživatelského rozhraní aplikace. To se vytvoří při inicializaci metodou CreateAppUiL(). Je to jediná metoda, kterou musí třída CAknDocument obsahovat. Ta tedy vytváří a vrací třídu uživatelského rozhraní dokumentu CPCH_v110AppUi, odvozenou od třídy CAknViewAppUi. Toto uživatelské rozhraní je zcela neviditelné. Pokračuje pouze ve vytváření tříd pohledů (view, container) aplikace odvozených od základní třídy CCoeControl. Tyto pohledy jsou již ovládacími prvky, starají se o vykreslování dat aplikace a interakci displeje s těmito daty.



Obrázek 7: Základní třídy pro implementace aplikací pod platformou series 60 (převzato z [22])

Na *obrázku 7* lze vidět, mimo vrstvy již zmíněné, vrstvy APPARC a CONE. První se stará o architekturu aplikace a druhá o ovládací prvky a jejich grafickou interakci. [20, 23]

Aplikace využívá dva pohledy. První ListBoxMAIN, který se zobrazí ihned po spuštění aplikace, zobrazuje seznam (List Box) s uloženými podmínkami (*obrázek 14*, *příloha 1*). Používá třídy: CListBoxMAINView a CListBoxMAIN. První je potomkem třídy CAknView z vrstvy uživatelského rozhraní Avkon. Tu vytváří třída CPCH_v110AppUi ve funkci InitializeContainersL() a slouží pro zobrazení uživatelských prvků pohledu. O interakci

s OS se stará druhá třída CListBoxMAIN, která je potomkem třídy CCoeControl. Ta se vytváří z třídy CListBoxMAINView při každé aktivaci pohledu ve funkci DoActivateL().

Druhý pohled SettingItemList slouží pro nastavování informací (definování podmínek a změny na čas), které pohled první zobrazuje (*obrázek 14, příloha 1*). Využívá třídy CSettingItemListView a CSettingItemList. První třída je potomkem opět známé třídy CAknView a vytváří ji opět třída CPCH_v110AppUi funkcí InitializeContainersL(). Ale druhá třída, protože pohled obsahuje speciální prvek pro nastavování informací (Setting Item List), je potomkem třídy CAknSettingItemList a vytváří ji funkce DoActivateL() třídy CSettingItemListView.

Třída CPCH_v110AppUi funkcí InitializeContainersL() pohledy pouze nevytváří, ale i registruje a identifikuje pohled, který je defaultní. Defaultní pohled je pohled ListBoxMAIN se seznamem uložených podmínek (*obrázek 14, příloha 1*). Každý pohled totiž vlastní jednoznačný identifikátor, které lze najít definované v souboru PCH_v110.hrh pomocí typu enum TPCH_v110ViewUids. Poté změna pohledu z jednoho na druhý probíhá pomocí funkce ActivateLocalViewL() ze třídy CAknAppUi vrstvy Avkon a tohoto jednoznačného identifikátoru.

Samotné pohledy a ovládací prvky na nich jsou převážně konstruovány z tzv. resources souborů. Každý pohled má svůj vlastní rssi soubor. Tyto ovládací prvky pak lze dynamicky upravovat, některé dokonce i vytvářet bez resource souboru. (více o tzv. resources jednotlivých pohledů v kapitolách *4.4.2* a *4.4.3*)

4.4.1 Vytváření tříd aplikačních pohledů

Všechny třídy aplikačních pohledů až na třídu CSettingItemListView musí být vytvářeny pomocí tzv. dvoufázové konstrukce za pomoci úklidového zásobníku CleanupStack. Je použita speciální funkce ConstructL(), do které je umístěna veškerá inicializace třídy, která by mohla skončit generováním leave (nedostatkem paměti). Jde o to umístit ukazatel na objekt do úklidového zásobníku pro případ, že by k tomuto došlo. Při vygenerování leave by poté objekt, který se inicializuje, byl jednoduše automaticky odstraněn pomocí tohoto úklidového zásobníku.

Pro zjednodušení a větší transparentnost existují šablony metod NewL() a NewLC(). Jedná se o statické metody, které lze volat bez jakékoliv existující instance třídy. Používají se následovně například při vytváření třídy CZ.

```
CZ* CZ::NewLC() {
    // alokace objektu CZ pomocí new(Eleave), při nedostatku paměti vyvolá výjimku leave
    CZ* self = new(Eleave) CZ;
    CleanupStack::PushL(self);          // jeho umístění do úklidového zásobníku
```

```
    self->ConstructL();                    // inicializace vytvořeného objektu CZ
    return self;
}

CZ* CZ::NewL() {
    CZ* self = CZ::NewLC();
    // pokud inicializace skončí v pořádku, z úklidového zásobníku je objekt vyjmut až zde
    CleanupStack::Pop();
    return self;
}
```

Metody `CZ::NewLC()` a `CZ::NewL()` fungují jako *funkce typu factory*, chovají se jako určitý druh konstruktoru.
[19]

4.4.2 Pohled pro zobrazení informací (ListBoxMAIN)

Resource soubor `ListBoxMAIN.rssi` definuje statické vlastnosti tohoto pohledu. Každý prvek v resource souboru má svůj identifikátor, pak jednotlivé prvky v něm můžou být těmito identifikátory i provázané. Například pomocí identifikátoru `r_list_boxmain` se identifikuje načtení celého pohledu ve funkci `ConstructL()` třídy `CListBoxMAINView`. Pod tímto identifikátorem je definice hlavního panelu ovládacích prvků „**`Options -- Hide`**“, a menu, které je spouštěno po stisku prvku „**`Options`**“ (*obrázek 14, příloha 1*). Definice menu udává jednotlivým tlačítkům jednoznačné identifikátory, které jsou také obsaženy v souboru `ListBoxMAIN.hrh` pomocí výčtu prvků `enum`. Tyto identifikátory jsou poté odchytávány ve funkci `HandleCommandL()` třídy `CListBoxMAINView`. Z tého funkce jsou podle identifikátoru, který byl odchytnut, volány funkce na obsluhu stisku jednotlivých tlačítek menu, které se nacházejí ve třídě `CListBoxMAIN`. V této třídě se také nachází funkce `OfferKeyEventL()`, která slouží pro odchytávání zrychlené volby, která je implementována pomocí jejího parametru `aKeyEvent`. Tento parametr identifikuje tlačítka klávesnice. Na *obrázku 14 přílohy 1.* lze vidět u každé položky menu číslo klávesnice, které slouží pro danou položku menu jako zrychlená volba.

Dále v resource souboru pohledu `ListBoxMAIN` je definice stavového řádku s názvem aplikace „**`Profile change`**“ a dialog, který slouží jako dotaz např. pro mazání podmínky (*obrázek 20 příloha 1*). Tento dialog je načítán z resource souboru a dále používán pomocí funkce `ShowQueryDialogL()`, která se nachází ve třídě `CListBoxMAIN`.

Seznam List Box, který lze vidět na *obrázku 14 přílohy 1.*, má přidanou tu vlastnost, že lze v něm označit více položek. Pomocí této vlastnosti uživatel může označit více uložených podmínek, které chce editovat či mazat. Tuto vlastnost definuje flag `EAknListBoxMarkableList`

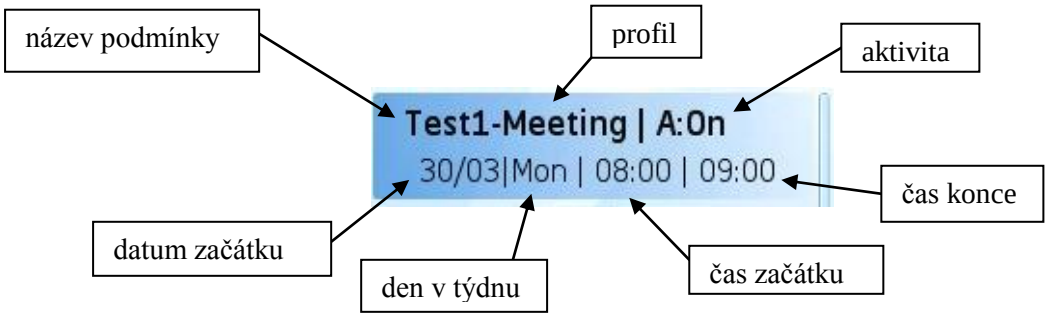
v definici samotného seznamu `LISTBOX` v resource souboru, který má identifikátor `r_list_boxmain_list_box`. Přímo v resource souboru lze definovat i statické prvky seznamu. V aplikaci „**`Profile change`**“ jsou ale přidávány až dynamicky za běhu ve funkci `InitizeListBox()` třídy `CListBoxMAIN`, která se spouští pokaždé, když je aktivován pohled (funkce `DoActivateL()`). Přidávány jsou tak definované informace, podle kterých se mění profily, aby byly přehledně viditelné. Způsob ukládání těchto informací bude dále vysvětlen v *kapitole 4.5*. Při případném nedostatku paměti na zobrazení všech informací v seznamu List Box, je rozhodnuto o vymazání nejzazších podmínek. Toto je jisté omezení, které je nutné, aby byla aplikace i poté použitelná. Toto se může stát například po vygenerování tolika podmínek, kdy na vygenerování je ještě paměť, ale na zobrazení už ne. Způsob detekce nedostatku paměti při generování/přidávání podmínek je popsán v *kapitole 4.5*. Seznam List Box je prakticky dynamické pole typu `CDesCArray`. Funkce `AppendL()` při přidávání do tohoto pole vygeneruje výjimku `leave KErrNoMemory`, a ta je odchytnuta makrem `TRAPD` až v již známé funkci `DoActivateL()` třídy `ListBoxMAINView`. Jednotlivé položky seznamu jsou typu Double item (*obrázek 8*). Při naplnění hodnotou typu string musí mít formát “`\tActivate\tvia Bluetooth\t`” (*obrázek 8*).

Obrázek 8: Double item prvek seznamu (převzato z [24])

Aplikace „**`Profile change`**“ prvky typu Double item používá následovně. Při aktivním nastavení změna na čas se zobrazuje:

Obrázek 9: Zobrazení aktivní změny na čas a popsané informace

U podmínek zobrazuje jejich parametry tímto způsobem:



Obrázek 10: Zobrazení parametrů podmínky

Hlavní funkce, která se stará o označení či odznačení jedné či více položek je `Mark()`. Nachází se ve třídě `CListBoxMAIN`. Jejím parametrem `item` se označuje číslo položky, která se má označit či odznačit. Při parametru `item = -1` se označují či odznačují položky všechny. Druhý její parametr `mark` typu `TBool` funkce udává, jestli se má položka označit či odznačit. Funkce projde všechny položky seznamu a podle tohoto parametru označí funkcí `SelectItemL()` nebo odznačí funkcí `DeselectItem()` všechny nebo určený prvek. Funkci `Mark()` používají jednotlivé funkce na označení. Všechny možnosti označení různých položek, lze najít v *návrhu aplikace* či v *příloze 1*.
[24]

4.4.3 Pohled pro nastavení informací (SettingItemList)

Statické vlastnosti tohoto pohledu definuje resource soubor `SettingItemList.rssi`. Pohled má identifikátor `r_setting_item_list_view` a je načítán opět ve funkci `ConstructL()` třídy `CSettingItemListView`.

Tento resource soubor opět definuje hlavní panel ovládacích prvků „**`Options -- Back`**“, a jednu položku v menu, které se otevírá stiskem „**`Options`**“. Této jedné položce v menu je později dynamicky upravován její popis podle toho pro jaké nastavení se pohled spouští. A při spuštění nastavení „**`Change on time`**“ se jedna položka menu „**`Next change`**“ dynamicky přidává. Dynamická úprava menu se prování ve funkci `DynInitMenuPaneL()` třídy `CSettingItemListView`. Tato třída tohoto pohledu má také funkci `HandleCommandL()`, která se stará o odchytávání tlačítek menu. Ty jsou opět odchytávány pomocí identifikátorů definovaných v souboru `SettingItemList.hrh`, které jsou přidruženy k příslušným tlačítkům v resource souboru.

Jako identifikátor pro to v jakém režimu je pohled spouštěn slouží proměnná `iSettingsItemListMode` typu `TInt` ze třídy `CPCH_v110AppUi`, které je přiřazován jeden z výčtu prvků `SettingsItemListMode` ze souboru `PCH_v110.hrh`. Vždy při změně na pohled pro nastavování informací, tedy před spuštěním funkce `ActivateLocalViewL()` s identifikátorem tohoto pohledu, je tato hodnota nastavena podle toho, jestli budeme podmínky:

- *generovat podmínky:* identifikátor `EModeGenerateConditions`
- *přidávat podmínku:* identifikátor `EModeAddCondition`
- *editovat podmínku/podmínky:* identifikátory `EModeEditOneCondition`, `EModeEditMoreConditionsWithT`, `EModeEditMoreConditionsWithA`, `EModeEditMoreConditionsWithW`, `EModeEditMoreConditionsWithAW`, `EModeEditMoreConditionsWithTA`, `EModeEditMoreConditionsWithTW`, `EModeEditMoreConditionsWithTAW`

Identifikátorů pro editaci je více, protože při editaci více podmínek naráz jsou editovány jen ty parametry z podmínek, které jsou stejné. Toto se před spuštěním pohledu pro editaci kontroluje a nastavuje se identifikátor podle této kontroly. Pomocí tohoto identifikátoru jsou později příslušné prvky nastavení zobrazeny či skryty.

- *nastavovat změnu na dobu:* identifikátor `EModeTimeChange`

Dále resource soubor tohoto pohledu definuje celý prvek Setting Item List s jednotlivými položkami nastavení. Tento Setting Item List je typu `AVKON_SETTING_ITEM_LIST` a má základní identifikátor `r_setting_item_list`. Je nadefinováno celkem deset prvků k nastavení celkem šesti typů. Jedná se o prvky:

- *Binary Switch:* Vrací typ `TBool` a v aplikaci slouží k nastavení, zda je/jsou podmínka/podmínky aktivní či neaktivní. Tento prvek spravuje třída `CAknBinaryPopup-SettingItem`.
- *Text Editor:* Obecně vrací typ specifikovaného deskriptoru (základní třída `TDesC`). Deskriptory jsou v Symbian OS použity k ukládání řetězců. Aplikace používá typ `TBuf`, což je prakticky obdobou `char[]` ze standardního jazyka C. V aplikaci typ `TBuf` slouží k nastavení jména. Prvek spravuje třída `CAknTextSettingItem`.
- *Enumerated Text:* Vrací typ `TInt` a v aplikaci slouží pro definování profilu, který je podmínkou/podmínkami měněn. Prvky v něm nejsou definovány pomocí resource souboru, ale jsou načítány dynamicky pomocí knihovny Profile Engine Wrapper (více o této knihovně v *kapitole 4.7*). Slouží k tomu třída `CMyTextPopupSettingItem`, která dědí původní třídu `CAknEnumeratedTextPopupSettingItem` vrstvy `AVKON`.
- *Time Editor:* Vrací typ `TTime` a v aplikaci slouží k definici startovního a koncového času při generování či přidávání podmínek. Typ `TTime` slouží pro ukládání data i času, ale prvek *Time Editor* do tohoto typu ukládá pouze čas. Symbian OS nemá typ, který by definoval pouze čas. Prvek spravuje třída `CAknTimeOrDateSettingItem`.
- *Slider:* Vrací typ `TInt` a v aplikaci slouží při generování podmínek k definování počtu dnů, po kterých má být podmínka opakována. Při nastavení změny na čas slouží pro definování počtu hodin a minut, po které má být profil nastaven. Prvek spravuje třída `CAknSliderSettingItem`.
- *Date Editor:* Vrací typ `TTime` a v aplikaci slouží k definování rozmezí startovního a koncového data při generování podmínek. Prvek spravuje třída `CAknTimeOrDate-SettingItem`, stejně jako u prvku *Time Editor*. Avšak *Date Editor* ukládá do typu `TTime` pouze datum.

Ovládací prvek setting list je načítán z resource souboru ve funkci `DoActivateL()` třídy `CSettingItemListView`. Každý z celkem deseti těchto prvků obsahuje jednoznačný

identifikátor, podle kterého jsou prvky vytvářeny, a k nim přidávány proměnné, do kterých vrací hodnoty při jejich nastavení. Prvky jsou vytvářeny funkcí `CreateSettingItemL()` třídy `CSettingItemList`, kterou musí tato třída minimálně jako jedinou obsahovat.

Identifikátory pro to, v jakém režimu je pohled spouštěn, jsou důležité i ve funkci `DoActivateL()`. Podle nich je totiž také rozhodováno, které z definovaných prvků budou skryty. Definují se při každé aktivaci pohledu všechny. Lze tak jeden pohled pro nastavení používat pokaždé s různými prvky. Zdálo se mi daleko efektivnější upravovat jeden pohled pro více použití, než definovat pro každé nastavení zvláštní pohled, když jsou u některých nastavení nutné stejné prvky. [24]

4.5 Způsob ukládání a vyhledávání informací

Způsob ukládání informací úzce souvisí s pohledem pro jejich nastavení `SettingItemList`, protože při stisku tlačítka z menu „**`Generate`**“, „**`Add`**“, „**`Edit`**“, „**`Next change`**“ a „**`Set`**“ se spouštějí příslušné funkce pro obsluhu těchto tlačítek. Jelikož jsou tlačítka „**`Generate`**“, „**`Add`**“ „**`Edit`**“ a „**`Set`**“ pouze přejmenované názvy tlačítka pro uložení (viz *kapitola 4.4.3*), tak se rozhodne, která funkce se použije pro jejich obsluhu, až ve funkci `HandleSaveMenu()` třídy `CSettingItemListView`. Rozhodne se tak opět podle identifikátoru pro to, v jakém režimu je pohled spouštěn.

Například při generování a přidávání podmínek se nejprve spouští funkce pro kontrolu správnosti zadaných údajů `CompareValuesAdd()`, a poté, pokud kontrola proběhne v pořádku, se spouští funkce pro jejich uložení `RunAddCondition()`. Obě patří do třídy `CSettingItemList`. Funkce podmínky/podmínku tvoří z proměnných, které jsou přidruženy k daným prvkům v nastavení. Všechny tyto proměnné jsou ve třídě `TSettings`. Jelikož prvky pro nastavení data a času v Setting Item List jsou zvlášť (viz *kapitola 4.4.3*) a OS Symbian nemá zvláštní typy pro datum a čas (má jen typ `TTime`, který obsahuje datum i čas dohromady), tak je nejprve vygenerován datum a k němu doplněn čas.

Vždy po konstrukci dané podmínky třídy typu `TCondition`, je podmínka uložena do pole `iConditions` typu `RArray` třídy `CPCH_v110AppUi`. Toto pole je dynamické a jde u něj také specifikovat hodnota tzv. granularity, to znamená, že například při `granularity=4`, což je uvedeno při inicializaci, je počáteční velikost pole čtyři prvky. Před vkládáním 5. prvku se kapacita pole navýší o další 4 prvky. Je proto u tohoto pole důležité správně ji nadefinovat, aby na jednu stranu se zbytečně neplýtvalo užitečným prostorem, ale na stranu druhou nedocházelo k realokaci skoro při každém vkládání prvku.

Jelikož při nastavení generování více podmínek jsou v nastavení rozmezí dat i roky, tak nastává problém s nedostatkem paměti pro podmínky. Nemá smysl omezovat toto nastavení tím, že by

z tohoto nastavení možnost zadání roků zmizelo. Uživatel by paměť mohl zaplnit i generování podmínek po měsících. Funkce `AppendL()` třídy `RArray`, pokud při přidávání prvku dojde paměť, vygeneruje výjimku `leave`. Pro odchytnutí této výjimky slouží makro `TRAPD`. Pokud k tomuto dojde, tak je aplikace vrácena do původního stavu, který byl před stiskem tlačítka „**Generate**“. Uživatel je na to upozorněn varovným hlášením. Má možnost upravit datum a pokusit se o generování znovu.

Pole s prvky `iConditions` je po dokončení vkládání prvků pokaždé seřazeno podle startovního času a data, tedy podle proměnné typu `TTime`. Slouží k tomu funkce `SortConditionsByTime()`. Ta nejprve definuje klíč třídy `TLinearOrder` pomocí funkce `CompareStartTime()` třídy `TCondition`. Tento klíč se poté použije jako parametr funkce `Sort()` na toto pole.

Pro ukládání nastavení změny na čas slouží dvě funkce z třídy `CSettingItemList`. První, při stisku tlačítka „**Next change**“, `RunNextProfileOnTime()` slouží pro průběžné ukládání nastavení do pole `iProfileOnTime`. Druhá, při stisku tlačítka „**Set**“, `RunSetProfileOnTime()` uloží poslední průběžné nastavení. Průběžně se ukládají hodnoty, které udávají jaký profil má být nastaven (typ `TInt`), a jak dlouho má být profil nastaven v hodinách (typ `TInt`) a v minutách (typ `TInt`). Až po uložení těchto průběžných hodnot tato funkce vypočítá konkrétní časy (typ `TTime`) změn profilů podle aktuálního času a uloží je k průběžným hodnotám do pole `iProfileOnTime`. Také nastaví proměnnou typu `TBool iProfileOnTimeActive` na `ETrue`. Podle této hodnoty se identifikuje, jestli je tato změna na čas aktivní či ne. U nastavení změny na čas probíhá také kontrola, zda uživatel nějaké hodnoty časů zadal pomocí funkce `ControlProfileOnTime()` třídy `CSettingItemList`.

Vždy po uložení informací aplikace aktivuje pohled pro zobrazení informací `ListBoxMAIN`. Ten při inicializaci ve funkci `DoActivateL()` třídy `CListBoxMAINView` při detekování informací dané informace pouze nezobrazí. V případě detekování uložených informací podle nich vybere a nastaví, případně přenastaví, generátor události pro aktivní objekt, který je popsán v následující *kapitole 4.6*.

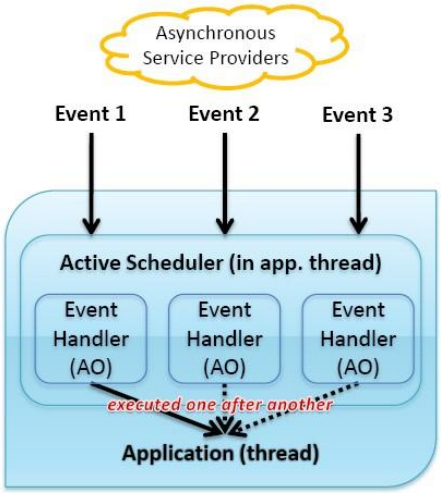
Toto vyhledání informací probíhá podle toho, jak je nastavena hodnota `iProfileOnTimeActive`. Pokud hodnota `iProfileOnTimeActive` je `ETrue`, stará se o obsluhu generátoru `RTimer` aktivního objektu funkce `SetProfileOnTime()`. Ta řídí automatické změny profilů pomocí nastavení změny na čas. Jelikož jsou již konkrétní časy změn profilů vypočítány a uloženy v poli `iProfileOnTime`, tak v tomto poli podle aktuálního času nalezne změnu nejbližší a na tuto generátor události nastaví.

Pokud hodnota `iProfileOnTimeActive` je `EFalse`, tak se stará o obsluhu funkce `ChooseEventandSetTimer()`. Ta řídí automatické změny profilů pomocí podmínek. Funkce

hledá nejbližší podmínku podle aktuálního času, podle které profil změní, v poli `iConditions`. Jelikož jsou v tomto poli podmínky seřazeny podle startovního času, tak se první dívá, zda není startovní čas větší než aktuální. Pokud je startovní čas větší než aktuální, tak funkce nalezla podmínku, která teprve bude startovat. Pokud není startovní čas větší než aktuální, tak se poté dívá, zda není koncový čas větší než aktuální. Pokud je koncový čas větší než aktuální a daná podmínka ještě neprobíhá, tak se nalezená podmínka také uskuteční. Funkce na nastavený profil změní ihned a pak nastaví časovač na čas konce podmínky. Funkce se taky dívá na aktivitu podmínky. Pokud podmínka není aktivní, tak se na změnách profilů neprojevuje.

4.6 Aktivní objekty

Důležitým prvkem v OS Symbian jsou aktivní objekty (*obrázek 11*). Jedná se o nepreemptivní multitasking. Aktivní objekty běží v kontextu jednoho vlákna. Aktivní plánovač rozhoduje podle priorit o pořadí provádění aktivních objektů. Při rozhodnutí o obsluze je volána funkce `RunL()` daného aktivního objektu a další rozhodnutí se provede, až funkce proběhne. Funkce `RunL()` tedy nemůže být nikdy přerušena. Pro práci s aktivními objekty slouží třída `CActive`. Při konstrukci aktivního objektu musí být aktivní objekt od této třídy odvozen.



Obrázek 11: Aktivní objekty OS Symbian (převzato z [22])

Aplikace „**`Profile change`**“ implementuje aktivní objekt v třídě `CAOchange` s generátorem události `RTimer`, což je třída časovače. Ta obsahuje funkci `At()`, která v daný čas, udaný jejím parametrem `aTime` typu `TTime`, vytvoří událost (spustí funkci `RunL()`) aktivního objektu. Ve funkci `RunL()` lze pak pomocí parametru `aStatus` typu `TRequestStatus` detekovat, zda daná událost proběhla v pořádku. Toto je hodně důležité, i když se to na první pohled nemusí zdát. Při testování na mobilním telefonu časy událostí končily v jiný než požadovaný čas. Dlouhým zkoumáním jsem zjistil, že je to díky tomu, že občas časovač vygeneruje událost se

statusem `aStatus = KErrAbort`, což knihovna popisuje jako zrušený časovač, protože se změnil systémový čas. [18]

Status `KErrAbort` je také využíván, pokud se opravdu změní systémový čas. Podle aktuálního času se poté rozhodne o novém nastavení generátoru událostí. Při aktivní změně na čas se použijí opět vypočítané časy z pole `iProfileOnTime`, pouze se kontroluje, zda aktuální nastavený čas již není v minulosti. Pokud je, tak se buď přejde na další profil, nebo se změna ukončí. Pokud systémový čas uživatel změní směrem dozadu, tak se změna na další profil provede podle času již vypočítaného (jen změnu oddálí). Pokud probíhá nějaká podmínka, tak se kontroluje znova, zda nový aktuální čas do ní spadá. Pokud ne tak je ukončena a je provedena změna na původní profil. Po této kontrole je opět volána funkce, která rozhodne o novém nastavení generátoru událostí.

Aplikace k funkci `At()` přistupuje pomocí funkce `StartProcess()` třídy aktivního objektu. Důležité jsou také ještě hodnoty proměnných `iWhichProfileOnTimeSet` a `iWhichConditionSet` typu `TInt` třídy `CAOchange`. Ty jsou nastaveny před spuštěním funkce `StartProcess()` s určitou hodnotou jejího parametru `aTimeToEvent` typu `TTime`, konkrétní nejbližší změny profilu, vybrané podle uložených informací. Tyto proměnné udávají pozici v poli uložených podmínek či nastavení změny na čas, na který prvek z pole byl nastaven časovač. Ve funkci `RunL()` se poté nejprve rozhodne podle aktivity jednoho či druhého nastavení, a dále podle těchto proměnných se na daný profil změní.

4.7 Profiles Engine Wrapper API

Aplikace používá tuto knihovnu pro načítání aktuálních profilů, změnu na jiný profil a k testu, zda daný profil ještě existuje. Tato knihovna je validní od OS Symbian verze 9.1. Knihovna potřebuje oprávnění tzv. Capability **`WriteDeviceData`** pro změnu na jiný profil, aby tato její funkce `SetActiveProfileL()` běžela na mobilním telefonu. Pro testování v emulátoru žádné oprávnění není třeba. Bez tohoto oprávnění aplikace lze přeložit a nainstalovat na konkrétní mobilní telefon. Pouze pokud má dojít ke změně na jiný profil, tak mobilní telefon hlásí varovnou hlášku, že aplikace nemá dostatečné oprávnění. (více o oprávněních Capability v *kapitole 4.9.2*)

Základem je třída `ProEngFactory`, pro vytvoření instance třídy `MProEngEngine` je třeba:

```
MProEngEngine* engine = ProEngFactory::NewEngineLC();
```

Další související třídy jsou na *obrázku 12*. Instance třídy `MProEngProfileNameArray` slouží pro získání pole aktuálních uložených profilů pomocí funkce `ProfileNameArrayLC()`.

```
MProEngProfileNameArray* iProfileArray(engine->ProfileNameArrayLC());
```

Funkce `ProfileId()` poté vrací, ke každému prvku z tohoto pole, jednoznačné `ID` uloženého profilu. Důležitá je také funkce `FindbyId()`, která vrací chybový kód `KErrNotFound` pokud

daný profil nebyl nalezen. Toto slouží k detekci profilu, který figuruje v aplikaci v nějakém nastavení, ale byl vymazán. Abychom se dostali ke jménu profilu, tak musíme ještě vytvořit instanci třídy `MProEngProfile`.

```
MProEngProfile* iProfile = engine->ProfileL(iProfileArray->ProfileId(i));
```

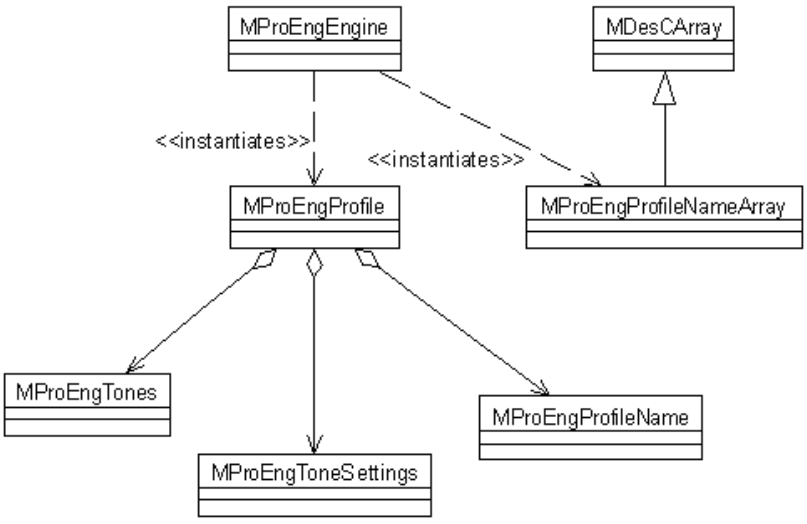
Jméno profilu spravuje až třída `MProEngProfileName`.

```
MProEngProfileName& iProfileName = iProfile->ProfileName();
```

Takto jsou získávány jména všech profilů v již zmíněné třídě `CMyTextPopupSettingItem` do prvku k nastavení *Enumerated Text* (viz. *kapitola 4.4.3*).

Funkce, které se starají o získání `ID` aktuálního nastaveného profilu či změnu aktuálního profilu pomocí jeho `ID` (`ActiveProfileId()` a `SetActiveProfileL()`) se nacházejí ve třídě `MProEngEngine`. Tyto `ID` jsou pro základní profily a nově přidané následující:

`0 = General` (výchozí), `1 = Silent`, `2 = Meeting`, `3 = Outdoor`, `4 = Pager`
`5 = Offline` – Tento profil se v emulátoru nachází, ale nelze na něj změnit. Emulátor hlásí „**`Unable to activate offline mode`**“.
`30-49` = toto rozmezí `ID` slouží pro profily vytvořené uživatelem
[25]



Obrázek 12: Třídy související s třídou MProEngProfile (převzato z [25])

4.8 Stream, úložiště

Aby uživatel po ukončení a znovu spuštění aplikace nemusel podmínky nastavovat znovu, tak se podmínky ukládají do souboru `Conf_condition.dat` pomocí souborového perzistentního úložiště třídy `CDirectFileStore`. Tento soubor[2] je na mobilním telefonu umístěn ve složce `C:/System/Apps/PCH_v110/`. Při jeho vymazání se aplikace chová jako nově nainstalovaná.

„Stream v systému Symbian je externí reprezentace jednoho nebo více objektů. Proces externalizace zahrnuje zápis dat objektu do streamu a opačný proces je nazýván internalizace.“ [19] Úložiště je množinou streamů. Jelikož je potřeba ukládat podmínky vždy všechny, a není potřeba měnit streamy poté, co jsou zapsány do úložiště, tak stačí použít třídu `CDirectFileStore`. Asociaci úložiště s aplikací zajišťuje identifikátor `UID` aplikace.

Ukládá se pouze nastavení podmínek, ne změny na čas. Při potřebě uložit všechny podmínky se volá funkce `SaveDataL()` třídy `CPCH_v110AppUi`. Tato funkce vytvoří soubor s podmínkami. Pokud již existuje, tak jej přepíše novým, a poté projde celé pole a pomocí operátoru << vkládá jednotlivé podmínky do streamu. Aby daný operátor fungoval, musí třída podmínky `TCondition` mít funkci `ExternalizeL()`.

Při spouštění aplikace se volá funkce `StartAplication()` třídy `CPCH_v110AppUi` a všechny podmínky jsou nyní pomocí operátoru >> a funkce `InternalizeL()` načteny zpět do pole podmínek `iConditions`.

4.9 Platform Security

Platform Security byla představena v OS Symbian verze 9. Jedná se o určité zapouzdření aplikačních dat a samotných aplikací. Vývojáři se tímto souborem bezpečnostních opatření musí řídit při vývoji aplikací.

4.9.1 Identifikátor UID

Každá aplikace pod OS Symbian musí mít jednoznačné označení pomocí identifikátoru `UID`, pomocí něj pak OS rozpoznává jednotlivé aplikace. Například se tak odlišují soubory asociované s aplikací. `UID` je to 32bitové číslo, které musí být v určitých intervalech podle potřeby použití. Aplikace „**`Profile change`**“ používá `UID` v rozsahu `0xE0000000-0xEfffffff` (OS v 9) pro testovací účely a tedy číslo `0xE7DBBE23`. Aplikace s `UID` v rozsahu pro testované účely nesmí být nikdy vydaná. [19]

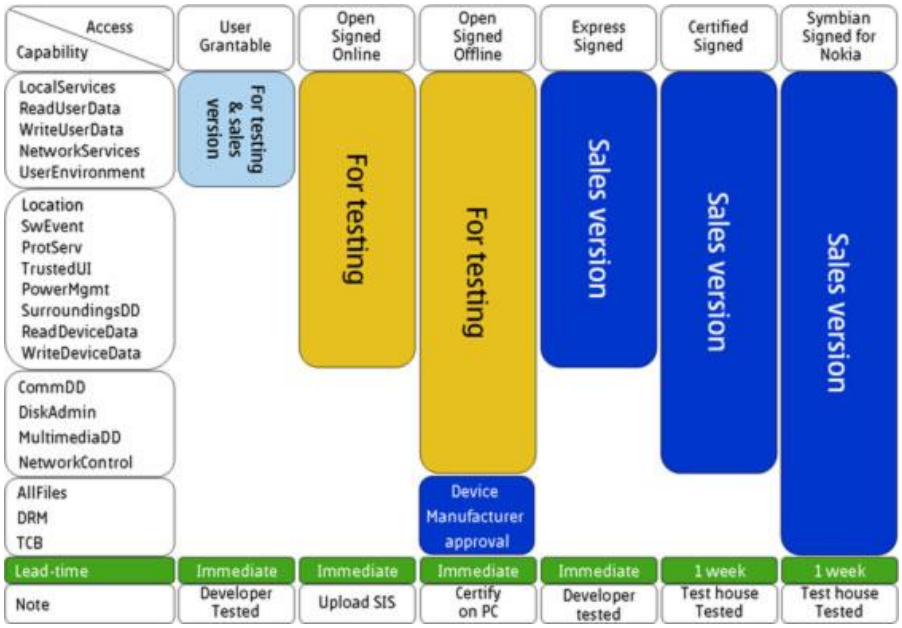
[2] Kde lze tento soubor najít v emulátoru, se nachází v příloze 2. nebo její textové verzi README.

4.9.2 Capability

Jedná se o přístupová práva k citlivým systémovým zdrojům, které se přidělují procesům (aplikacím). Aplikace „**`Profile change`**“ potřebuje díky použité knihovně `Profile Engine Wrapper API`, která je popsána v *kapitole 4.7*, oprávnění k přístupu **`WriteDeviceData`**. Opravňuje k zápisu do citlivých systémových dat. Toto nastavení ovlivňuje chování přístroje. [26]

Oprávnění Capability jsou poskytovány podle citlivosti přístupu. *Obrázek 13* popisuje možnost získání všech 20 oprávnění Capability. Nejcitlivější oprávnění *(***`AllFiles, DRM, TCB`***)* jsou poskytovány pouze po procesu schválení (nezávislé testování), nepomůže ani identifikátor `UID` z oblasti pro testovací účely.

Aplikace je touto schopností Capability označena v souboru `mmp` pomocí návěští `CAPABILITY` a za ním mezerou oddělenými názvy všech potřebných oprávnění. Toto bohužel nestačí, aby aplikace správně fungovala v mobilním telefonu. Aplikaci je nutno ještě digitálně podepsat jedním ze způsobů popsaných v *kapitole 4.9.4*. Prakticky až procesem podepsání jsou tato oprávnění přidělena.



Obrázek 13: Capability Granting Process (převzato z [26])

4.9.3 Data Caging

Jedná se o to, že jsou soubory aplikace uloženy ve svých privátních adresářích, a jiná aplikace nemůže přistupovat k souborům aplikace druhé. Také systémové soubory operačního systému jsou pro běžné aplikace skryty. Pouze aplikace s oprávněním Capability `TCB` smí přistupovat do celého souborového systému. Dále spustitelné soubory se musí nacházet v systémovém adresáři `*/sys/bin/`, jinak jich nelze spustit. [27]

Spustitelný soubor aplikace „**Profile change**“ je `PCH_v110.exe`, jak lze vyčíst ze souboru `PCH_v110.pkg`, který popisuje instalaci.

4.9.4 Symbian Signed

Jedná se o nejdůležitější část Platform Security. Tímto procesem je do aplikace zakódován digitální certifikát. Je nutno dodat, že je podepisován až vytvořený instalační soubor `sis`. Certifikát udává samozřejmě původ aplikace, ale co je důležitější, povoluje přístup k právům Capability.

Nyní umožňuje Symbian Signed tyto způsoby podepsání:

- *Open Signed – Online*: Jednoduché podepsání aplikace zdarma, pro testovací účely nebo osobní použití. Omezením je to, že při procesu se udává `IMEI` kód telefonu a aplikace se pak dá nainstalovat pouze na daný telefon. Aplikace je také podepsána pouze na 36 měsíců. Pro tento způsob podepsání slouží url:
  https://www.symbiansigned.com/app/page/public/openSignedOnline.do
  Bohužel díky velkému zájmu je tato www stránka občas nedostupná.
- *Open Signed – Offline*: Tento způsob podepsání již není tak jednoduchý, vývojář musí vlastnit Publisher ID, které není zdarma, a musí mít účet u Symbian Signed. Až toto splní, tak může požadovat Developer Certificate.S ním může podepsat aplikaci až pro 1000 zařízení. Ale musí být známy `IMEI` kódy zařízení, na které bude daná aplikace instalována. Tento Developer Certificate je platný 36 měsíců.
- *Express Signed*: Jedná se o způsob podepsání, které je možné i pro komerční nasazení programu. Není omezený na `IMEI` kódy telefonu. Aplikace jsou podepsány na 10 let a musí splňovat Symbian Signed Test Criteria.
- *Certified Signed*: Toto podepsání aplikace je možné pouze pro komerční účely. Taková aplikace musí být nezávisle otestována, a to není zdarma. Aplikace poté může mít oprávnění i na nejcitlivější Capability a může se pyšnit logem „*for Symbian OS*“.

[28]

Existují dokonce knihovny API, které fungují pouze po důvěryhodném podepsání jednou z těchto možností. Narazil jsem tak na knihovnu, kterou používám pro automatický start aplikace `Startup List Management API`. Ta proto funguje až v samotném mobilním telefonu, jelikož se podepisuje až instalační soubor `sis`.

Aplikaci „**Profile change**“ jsem pro testování na mobilním telefonu podepisoval pomocí *Open Signed – Online*. Popis tohoto podepsání se nachází v *příloze č. 2*.

5 Testování

Po implementaci aplikace proběhlo testování. Nejprve byla aplikace testována pomocí emulátoru EPOC, který je součástí vývojového balíčku SDK. Emulátor je po přeložení zdrojového kódu aplikace spouštěn z vývojového nástroje Carbide.c++. Zde byly odladěny prvotní chyby implementace. Až se aplikace zdála být funkční a použitelná v emulátoru, byla testována na reálném telefonu Nokia N95 (Symbian OS v 9.2).

Na reálném telefonu byla aplikace podrobena řadě testů, které by mohly nastat v reálném provozu. V testech se ověřuje funkčnost generování, přidávání, editace a mazání podmínek a dále nastavování a rušení nastavení změny na čas. Také se testuje správnost změny profilů, jak se aplikace zachová při přidání či vymazání profilu, jak se aplikace zachová při změně času v systému a jak se program zachová při editaci či mazání právě probíhající podmínky. I když tyto funkce můžou mít variant více, uveden je vždy u každé funkce příklad testu (uživatelského scénáře), jaký by mohl nastat.

1. ***Generování podmínek*** – Aktuální čas: 23:09, 27.4.2009

   Uživatel pomocí menu „**Options->Define condition(s)->Generate**" spustí pohled pro nastavení těchto hodnot.:

Condition name:	Test1
Profile:	Silent
Time change:	14:00
Time change back:	16:55
After how many days:	7
Start date:	20.4.2009
End date:	30.6.2009

   Po nastavení hodnot stiskne tlačítko „**Options->Generate**".

   Jelikož první podmínka by byla data 20.4.2009 se tak generování neprovede, zobrazí se varovné hlášení „**Start date is long ago in past.**". Uživatel je nucen posunout datum startu generování. Toto je z důvodu, že se staré podmínky mažou. Zobrazí se, pokud by měla být jedna z vygenerovaných podmínek takto smazána. Po upravení startovního data na 23.4.2009 a opětovného stisku generování je zobrazeno informativní hlášení „**10 new conditions have been generated**". Je vygenerováno 10 podmínek s daty 23.4., 30.4., 7.5., 14.5., 21.5., 28.5., 4.6., 11.6., 18.6. a 25.6.

2. ***Generování podmínek*** – Aktuální čas 23:16, 27.4.2009

   Uživatel pomocí klávesové zkratky [1] spustí pohled pro nastavení:

Condition name:	Test1

Profile:	Meeting
Time change:	16:00
Time change back:	3:00
After how many days:	1
Start date:	27.4.2009
End date:	15.6.2014

Jelikož je název této množiny podmínek stejný jako množiny generované dříve, tak je zobrazeno varovné hlášení „**This name is already used.**“. Změníme jej na Test2. Jelikož tato generovaná množina podmínek bude překrývat podmínky z množiny Test1, tak se zobrazí varovné hlášení „**One of generated conditions overlap conditions saved. [Test1, 30.4. 14:00]**”. Po změně startovního času na 23:00 již žádné kontroly této množině generovaných podmínek nevadí. Jelikož jsou ale generovány denně, tak na tolik podmínek v rozmezí 5 let nebude v telefonu již paměť. Zobrazí se varovné hlášení „**Memory is full. Can't add so many conditions.**”. Po změně na koncové datum 15.6.2013 již na vygenerování 1511 podmínek paměť bude. Po vygenerování se při zobrazení pohledu tento počet zredukuje o nejzazší podmínky díky tomu, že je potřeba ještě paměť na zobrazení těchto podmínek. V tomto případě zůstane podmínek 1210. Toto je bohužel jisté omezení. Popsáno je v *kapitole 4.4.2*. Také se ihned přejde na profil Meeting, jelikož aktuální čas spadá do podmínky vygenerované na den 27.4.2009.

3. ***Nastavení změny na čas*** – Aktuální čas: 23:24, 27.4.2009

   Uživatel pomocí klávesové zkratky [5] spustí pohled pro toto nastavení. Jelikož jedna podmínka probíhá tak se zobrazí otázka „**One condition is running cancel it?**“. Po stisku „**Yes**“ ji zruší a ihned se přejde na původní profil před rušenou podmínkou. Dále se spustí se pohled pro nastavení:

   (další nastavení volí stiskem „**Options->Next change**“)

Profile	Silent	Meeting	Outdoor
Duration-hours	2	1	0
Duration-minutes	30	3	2

   Po stisku „**Options->Set**” dojde k nastavení a spuštění změny na čas. Zobrazí otázka „**Make all conditions inactive. Continue?**“ Po stisku „**Yes**“ se zobrazí „**Time change has been set.**“ a ihned se přejde na profil Silent. Toto nastavení změny na čas bude aktivní při dalších testech.

4. ***Vymazání podmínek*** – Aktuální čas: `23:28, 27.4.2009`

   Uživatel označí pomocí klávesové zkratky `[8]`[3] podmínky z množiny `Test2` s dnem v týdnu ve čtvrtek a v pátek. Stiskem tlačítka `[c]` se zobrazí otázka „**`Delete total 346 conditions?`**‘. Stiskem „**`Yes`**“ je smaže. Zobrazí se hlášení „**`346 conditions deleted.`**“.

5. ***Editace podmínek*** – Aktuální čas: `23:32, 27.4.2009`

   Označení úterních a středečních podmínek z množiny `Test2` klávesovou zkratkou `[8]`. Po stisku klávesové zkratky `[3]` se nejprve zobrazí otázka „**`Can’t edit parameter active until Time change is active. Continue?`**“. Po stisku „**`Yes`**” proběhne editace parametrů na:

`Profile:`	`Outdoor`
`Time change:`	`23:30`
`Time change back:`	`3:02`

   Stiskem „**`Options->Edit`**” se zobrazí „**`346 conditions have been edited.`**“.

6. ***Editace podmínek*** – Aktuální čas: `23:39, 27.4.2009` (stále aktivní změna na čas)

   Stiskem středového tlačítka joysticku uživatel označí dvě podmínky s parametry:

`Active:`	`Off`	`Off`
`Conditions name:`	`Test2`	`Test2`
`Profil:`	`Outdoor`	`Meeting`
`Time change:`	`23:30`	`23:00`
`Time change back:`	`03:02`	`03:00`
`Start date:`	`29.4. (We)`	`2.5. (Sa)`

   Jelikož je stále aktivní změna na čas, tak aktivita podmínek editovat nelze, stejně tak startovní datum podmínek lze editovat, pouze pokud editujeme podmínku samotnou.

   Jelikož jsou ostatní parametry odlišné. Aplikace zobrazí varovné hlášení: „**`Marked conditions haven’t same parameter!`**“.

7. ***Zrušení změny na dobu*** – Aktuální čas: `23:41, 27.4.2009`

   Uživatel pomocí menu „**`Options->Time change->Cancel`**” změnu na čas zruší. Na původní profil před touto změnou se přejde ihned. Jelikož aktuální čas opět spadá do podmínky `Test2` s datem `27.4.2009`, tak se přejde na profil `Meeting`.

8. ***Přidání podmínky*** – Aktuální čas: `23:44, 27.4.2009`

   Uživatel pomocí menu: „**`Options->Define conditions->Add`**” spustí nastavení:

`Condition name:`	`Test3`
`Profile:`	`Pager`

[3] Označuje všechny podmínky ze stejné množiny, které mají datum ve stejný den v týdnu.

Time change:	11:00
Time change back:	14:55
Start date:	29.4.2009

Po stisku „**Options->Add**" je daná podmínka přidána. Zobrazí se hlášení „**1 new condition has been added.**".

9. ***Editace podmínky*** – Aktuální čas: 23:50, 27.4.2009

   Uživatel podmínku Test3 data 29.4.2009 označí středovým tlačítkem joysticku a zrychlenou volbou [3] edituje. Změní na hodnoty:

Active:	Off
Profile:	Pager
Time change:	11:00
Time change back:	14:55
Start date:	29.5.2009

10. ***Změna času v systému*** – Na aktuální čas: 12:03, 29.5.2009

    Uživatel pomocí nastavení telefonu, mění systémový čas. Nový aktuální čas spadá do podmínky Test3. Jelikož je tato podmínka neaktivní, tak se na změně neprojevuje.

11. ***Editace neaktivní podmínky*** – Aktuální čas: 12:05, 29.5.2009

    Po změně aktivity podmínky Test3 na On se po dokončení editace ihned změní profil na Pager. Díky předchozí změně systémového času a novým načtením seznamu List Box také došlo k odstranění podmínek, které jsou déle než 7 dní staré.

12. ***Změna času v systému*** – Na aktuální čas: 15:30, 29.5.2009

    Jelikož již aktuální čas do podmínky Test3 nespadá, tak je profil změněn na původní.

13. ***Změna času v systému*** – Na aktuální čas: 22:56, 1.6.2009

    Žádná podmínka nespadá do aktuálního času. Nic se neděje.

14. ***Změna profilu*** – Aktuální čas: 23:00, 1.6.2009

    Změna profilu na Meeting ve startovní čas podle podmínky Test2.

15. ***Editace podmínek*** – Aktuální čas: 23:05, 1.6.2009

    Označení podmínek z množiny Test2 dne 31.5.2009 a 1.6.2009. Při pokusu o editaci se zobrazí varovné hlášení „**Can't edit condition which is running.**".

16. ***Mazání podmínek*** – Aktuální čas: 23:07, 1.6.2009

    Označení všech podmínek množiny Test2 pomocí rychlé volby [7]. Při stisku tlačítka menu „**Options->Define condition(s)->Delete**" se zobrazí otázka „**Delete total 847 conditions?**". Po stisku tlačítka „**Yes**" se zobrazí další otázka „**Delete condition which is running?**". Po stisku tlačítka „**No**" se zobrazí informativní

hlášení „**846 conditions deleted.**“. Zůstala v paměti z množiny `Test2` pouze podmínka, která probíhá. Jelikož od poslední změny systémového času došlo k novému načtení seznamu List Box až nyní, tak se také odstranily z ostatních množin podmínky, které jsou staré déle než 7 dní.

17. ***Mazání podmínek*** – Aktuální čas: `23:14, 1.6.2009`

    Označení podmínek `Test2 1.6.` a `Test3 29.5.` Při stisku tlačítka vymazat se opět zobrazí otázka, zda se má vymazat probíhající podmínka. Při stisku ano se vymaže i tato podmínka. Na původní profil, ze kterého měnila, se poté přejde ihned.

18. ***Přejmenování profilu*** – Aktuální čas: `23:14, 1.6.2009`

    Uživatel přejmenuje profil `Silent` na `Silent-new`. Tento profil používají podmínky z množiny `Test1`. Při znovu načtení seznamu List Box se tato změna projeví. Pokud je aplikace pouze skrytá a uživatel ji pouze přesune na popředí tak k znovu načtení seznamu List Box nedochází.

19. ***Vymazání profilu*** – Aktuální čas: `23:14, 1.6.2009`

    Uživatel vytvoří nový vyzváněcí profil v OS. Edituje na něj podmínku z množiny `Test1 4.6.2009`. Následně nově vytvořený profil ze systému vymaže. Při změně systémového času na tuto podmínku dochází ke změně na profil výchozí (`General`). V seznamu List Box se tato změna projeví až po jeho znovu načtení.

Závěr

Seznámil jsem se s nejvíce rozšířenými operačními systémy v mobilních telefonech. Jelikož tyto patří do obecnější oblasti mobilních zařízení, částečně jsem se seznámil i s touto oblastí. Zaměřil jsem se především na operační systém Symbian. Pomocí těchto znalostí došlo k návrhu aplikace „**`Profile change`**“, která inteligentně mění vyzváněcí profily.

Při návrhu aplikace jsem kladl důraz na to, aby uživateli práce s programem a jeho nastavením nezabrala moc času. Výhodou je rychlé generování a editace potřebných podmínek, které pokryjí změny profilů na dané období. Tyto změny jsou pro události, které uživatel ví dopředu. Jelikož uživatel mnohdy potřebuje i změnu okamžitou, ale jen na určitý čas, tak byla navržena i možnost tohoto nastavení. Aplikace má poměrně složité uživatelské rozhraní, pomocí kterého dochází k nastavením. To jsem se snažil navrhnout a implementovat tak, aby bylo co nejvíce intuitivní pro uživatele. Danou aplikaci jsem pomocí dostupných knihoven implementoval pod verzí OS Symbian verze 9.2.

Při implementaci jsem hojně využíval dostupné fórum a Wiki knihovnu společnosti Nokia. Bez nich jen pomocí dostupné literatury by implementace byla podstatně složitější a náročnější. Princip tvorby aplikací s uživatelským rozhraním pod OS Symbian je poměrně odlišný od OS Windows či Linux. Proto jsou pro vývojáře, který s tímto systémem začíná, velmi užitečné ukázky použití jednotlivých knihoven či grafických prvků. OS Symbian má několik set výjimek tzv. panic, rozdělené do šestnácti kategorií. Emulátor `EPOC` upozorňuje na chyby implementace ukončením vlákna s aplikací a zobrazením dané kategorie a čísla výjimky. Již pomocí tohoto lze identifikovat případnou chybu, pokud ani toto nepomůže, tak emulátor má poměrně přehledný debugger.

Testoval jsem jak v emulátoru, tak na reálném mobilním telefonu. Toto testování pokrývalo všechny možné uživatelské scénáře, které by mohly nastat. Díky tomuto testování se domnívám, že je aplikace použitelná a schopná reálného provozu.

I když nastavením aplikace lze jednoduše pokrýt libovolné období pro automatické změny profilů, tak případný další rozvoj aplikace by mohl spočívat především v dalších možnostech nastavení. Pro uživatele, kteří hojně využívají kalendář, by mohlo být atraktivní měnit profily podle podmínek uložených v kalendáři. Tuto možnost nastavení jsem si zpočátku nevybral, protože ne všichni uživatelé kalendář v telefonu využívají. Dále například nastavení aplikace by mohlo být interaktivně spravováno pomocí programu v počítači. Aplikaci prezentuje plakát, který je *přílohou*.

Literatura

[1] Epocware – Paragon Software Group. *Handy Profiles for Nokia N95* [online]. [cit. 22.4.2009]. Dostupné na URL: <http://nokia-n95-software.epocware.com>

[2] Nova Mobility Systems. *Nova's Side Arm 2 UMPC* [online]. [cit. 2009-4-10]. Dostupné na URL: <http://www.novamobility.com>

[3] HTC Corporation. *HTC Dream Overview* [online]. [cit. 2009-4-10]. Dostupné na URL: <http://www.htc.com>

[4] Wikimedia Foundation, inc. *Wikipedia: Smartphone* [online]. [rev. 5. April, 2009]. [cit. 2009-4-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/Smartphone>

[5] Wikimedia Foundation, inc. *Wikipedia: Palm OS* [online]. [rev. 4. April, 2009]. [cit. 2009-4-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/Palm_OS>

[6] Palm, Inc. *Palm Developer Guide, Palm OS Platform*. [rev. October 10, 2007]. [cit. 2009-4-10] Dostupné na URL: <https://pdnet.palm.com>

[7] GNU Free Documentation License, *Openmokowiki: Openmoko* [online]. [rev. 5. March, 2009]. [cit. 2009-4-10]. Dostupné na URL: <http://wiki.openmoko.org/wiki/Main_Page>

[8] *Android Developers* [online]. [cit. 2009-4-10]. Dostupné na URL: <http://developer.android.com/>

[9] Wikimedia Foundation, inc. *Wikipedia: Android (operating system)* [online]. [rev. 5. April, 2009]. [cit. 2009-4-10]. Dostupné na URL: <http://en.wikipedia.org>

[10] LiMo Foundation. *LimoFoundation: What is the Platform* [online]. [cit. 2009-4-10]. Dostupné na URL: <http://www.limofoundation.org>

[11] Wikimedia Foundation, inc. *Wikipedia: Limo Platform* [online]. [rev. 21. March, 2009]. [cit. 2009-4-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/LiMo_Platform>

[12] Research In Motion Limited, *BlackBerry Support & Services Product Documentation* [online]. [cit. 2009-5-2]. Dostupné na URL: <http://na.blackberry.com/eng/support/docs/>

[13] Apple Inc. *iPhone DevCenter* [online]. [cit. 2009-4-10]. Dostupné na URL: <http://developer.apple.com/iphone/>

[14] Wikimedia Foundation, inc. *Wikipedia: Windows Mobile* [online]. [rev. 5. March, 2009]. [cit. 2009-4-10].
Dostupné na URL: <http://en.wikipedia.org/wiki/Windows_Mobile>

[15] Microsoft Corporation. *Windows Mobile Developer Center* [online]. [cit. 2009-4-10].
Dostupné na URL: <http://msdn.microsoft.com>

[16] Wikimedia Foundation, inc. *Wikipedia: Symbian OS* [online]. [rev. 4. March, 2009]. [cit. 2009-4-10].
Dostupné na URL: <http://en.wikipedia.org/wiki/Symbian_OS>

[17] Symbian. *Symbian Developer Network* [online]. [cit. 2009-4-10].
Dostupné na URL: <http://developer.symbian.com>

[18] Symbian Software Ltd. *Symbian OS v9.2 Developer Library* [online]. [cit. 2009-3-23].
Dostupné na URL: <http://www.symbian.com>

[19] HARRISON, Richard. *Programujeme aplikace Symbian OS v jazyce C++*. Brno: Computer Press, 2006. ISBN 80-251-1243-8

[20] HARRISON, Richard. *Symbian OS C++ for mobile phones.Volume 2, Programming with extended functionality and advanced features*. Chichester: John Wiley & Sons, 2004. ISBN 0-470-87108-3

[21] COULTON, Paul, EDWARDS, Reuben. *S60 Programming a tutorial guide*. Chichester: John Wiley & Sons. 2007. ISBN 978-0-470-02765-3

[22] JAKL, Andreas. *Symbian resources: training material for Symbian OS C++ development* [online]. Austria: University of Applied Sciences in Hagenberg, 2008.
Dostupné na URL: <http://www.symbianresources.com/tutorials/>

[23] BABIN, Steve. *Developing Software for Symbian OS, An Introduction to Creating Smartphone Application in C++*, Chichester: John Wiley & Sons, 2006. ISBN 0-470018-45-3

[24] Nokia Corporation. *S60 Platform: Avkon UI Resources* [online]. [cit. 2009-4-5].
Dostupné na URL: <http://www.forum.nokia.com>

[25] Nokia Corporation. *S60 3rd Edition C++ Developer's Library v1.0: Profiles Engine Wrapper API* [online]. [cit. 2009-4-5].
Dostupné na URL: <http://library.forum.nokia.com>

[26] Nokia, *Forum Nokia: Capability descriptions* [online]. [cit. 2009-3-30].
Dostupné na URL: <http://www.forum.nokia.com>

[27] Nokia. *Wiki Forum Nokia: Data Caging* [online]. [cit. 2009-4-4].
Dostupné na URL: <http://wiki.forum.nokia.com>

[28] MORRIS, Ben. *A guide to Symbian signed*. Symbian Software Ltd.
Dostupné na URL: <http://developer.symbian.com>

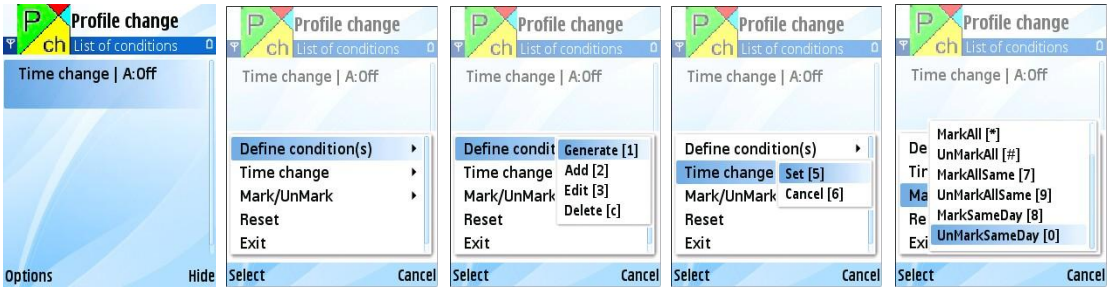
Seznam příloh

Příloha 1 –

Nápověda k aplikaci `Profile Change`

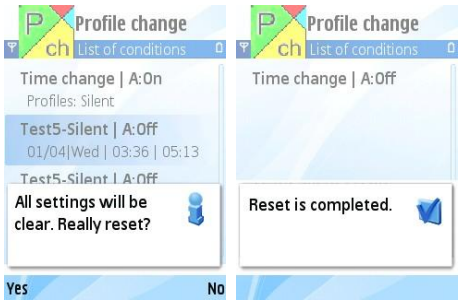
1. Základní funkce

Aplikaci lze po úspěšném nainstalování najít v menu OS Symbian ve složce Installed. Jinak se aplikace automaticky spouští po startu systému. Aplikaci lze skrýt, aby běžela na pozadí stiskem tlačítka „**`Hide`**“ z panelu ovládacích prvků (*obrázek 14*).



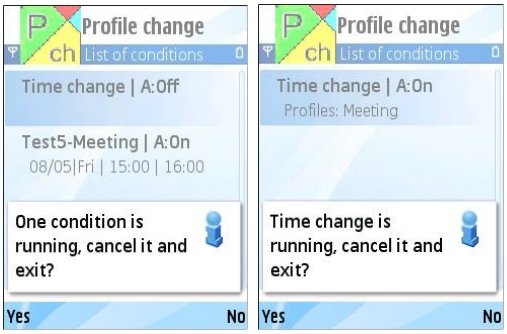
Obrázek 14: Spuštěná aplikace v popředí bez nastavení, menu aplikace, kaskádové menu pro definování podmínky a změnit na dobu, menu Mark/UnMark

Aplikace inteligentně spravuje vyzváněcí profily podle daných nastavení, které se nastavují pomocí menu. Menu se spouští pomocí druhého tlačítka „**`Options`**“ z panelu ovládacích prvků. Nastavení jsou dvojího typu. Lze definovat neomezené množství podmínek „**`Define conditions(s)`**”. Pomocí tohoto nastavení lze pokrýt automatické změny profilů za určité období, den v týdnu a čas. Toto nastavení je určené především na události známé dopředu. Dále aplikace umožňuje nastavit tzv. změnu na čas „**`Time change`**“ až pro 3 profily, které se postupně změní. Druhé nastavení je především pro události, které se vyskytnou náhle. Podmínky lze generovat, přidávat, editovat a mazat. Změna lze pouze nastavit a zrušit. To lze jak z menu, tak pomocí klávesových zkratek, které jsou vidět na *obrázku 14*. V menu se také nachází položka „**`Reset`**“, která slouží pro vymazání všech nastavení. Aplikace se poté chová jako nově nainstalovaná (*obrázek 14*).



Obrázek 15: Spuštění resetu aplikace

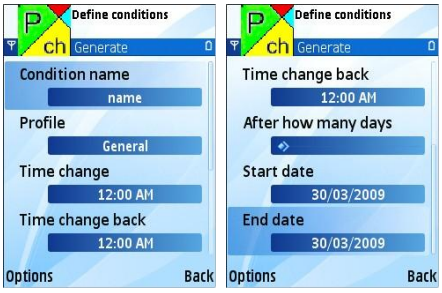
Při stisku tlačítka „**Konec**“ se aplikace ukončí a nastavení podmínek se ukládá. Neukládá se nastavení změna na čas a změny podmínek, které provedla (jejich zneaktivnění). Při ukončení aplikace se již žádné profily nemění. Pokud při stisku tlačítka „**Konec**“ probíhá změna na čas či nějaká z podmínek tak je na to uživatel upozorněn (*obrázek 16*).



Obrázek 16: Upozornění při stisku tlačítka Konec

2. Generování, přidávání podmínek

Při generování množiny podmínek se stejným jménem se zadávají parametry, jména podmínek „**Condition name**“, profil[4] „**Profile**“ na, který se danými podmínkami bude měnit, čas změny „**Time change**“, kdy se každá podmínka uskuteční, čas změny zpět „**Time change back**“, kdy se uskuteční změna na původní profil před změnou. Tyto dva časy udávají rozmezí, mezi kterýma je aplikace schopna na daný profil změnit. Pokud je například aktuální čas uprostřed momentálně vygenerované podmínky, tak změna na daný profil také proběhne. Dále se zadává, po kolika dnech se bude daná podmínka generovat „**After how many days**“, a „**Start date**“ a „**End date**“ udává rozmezí dvou dat, mezi kterými generování proběhne. Toto nastavení lze vidět na *obrázku 17*.



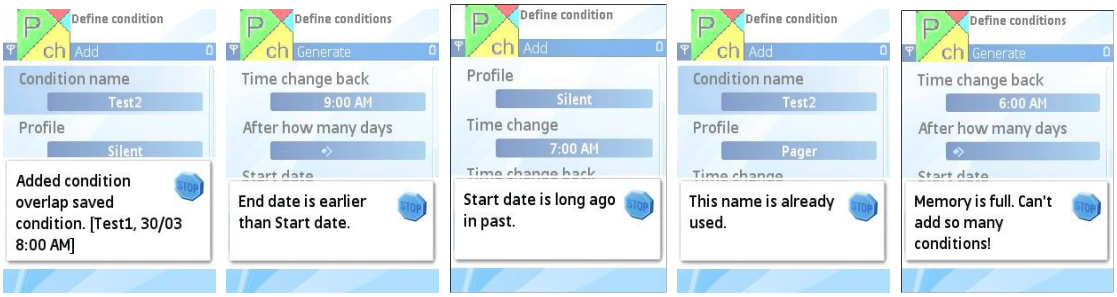
Obrázek 17: Nastavení generování množiny podmínek

První vygenerovaná podmínka z množiny bude data, které udává startovní datum generování a od tohoto data v rozmezí, které uživatel zadá pomocí „**After how many days**“ (rozmezí 1 až 7 dní) se vygenerují ostatní podmínky z množiny (minimálně jedna v startovní datum). Při volbě přidání jedné podmínky zmizí nastavení „**After how many days**“ a „**End date**“. Jedna podmínka se

[4] Aplikace načítá všechny aktuální profily, jak výchozí, tak nově přidané. Pokud před změnou je profil smazán, tak se přejde na první výchozí profil (Normální, General).

vygeneruje v daný startovní den „**Start date**“. Pokud uživatel zadá čas změny zpět „**Time change back**“ menší než čas změny „**Time change**“, tak se změna zpět provede až následující den.

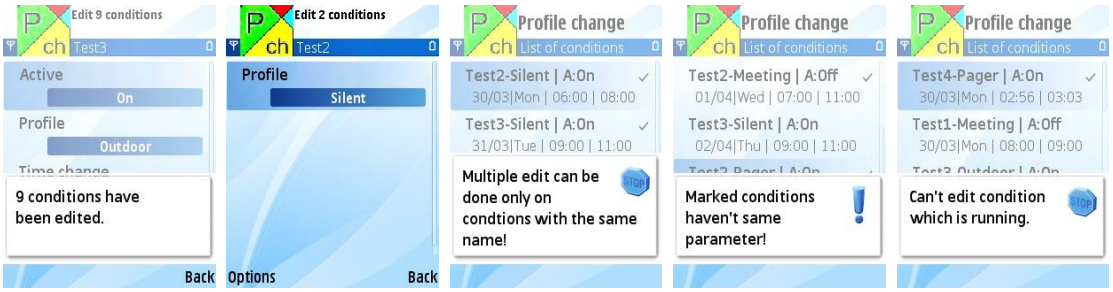
Při generování se kontroluje, jestli vygenerovaná množina nepřekryje některou z podmínek již uložených. Tato kontrola probíhá také při přidávání podmínky samotné. Pokud překrývá, tak je na to uživatel upozorněn varovným hlášením (*obrázek 18*). Dále je kontrolováno při generování množiny podmínek rozmezí generování, jestli je startovní datum větší než koncové datum (*obrázek 18*). Také se kontroluje startovní datum u generovaných podmínek, zda jej uživatel nezadal tak nízký, že by se po vygenerování podmínky zrovna smazaly (podmínky staré 7 dní se automaticky mažou). Dále se kontroluje, jestli se již nevyskytuje stejné jméno podmínky u množiny vygenerovaných podmínek, nebo u samostatně přidané jedné podmínky (*obrázek 18*). Po zobrazení těchto varovných hlášení se uložení neprovede. Uživatel je nucen nastavení opravit a pokusit se provést generování či uložení znovu pomocí „**Options**“ **->** „**Generate**”. Nově přidané či vygenerované podmínky se ukládají jako aktivní. Tato aktivita lze změnit editací. Poslední kontrola spočívá v tom, pokud by uživatel chtěl vygenerovat více podmínek, než na které je paměť (*obrázek 18*).



Obrázek 18: Varovná hlášení při generování či přidávání

3. Editace podmínek

Při editaci jedné podmínky lze editovat: jestli je aktivní „**Active**“ (aktivita udává, zda se bude podmínka na změnách projevovat), profil „**Profile**“, čas změny „**Time change**“, čas změny zpět „**Time change back**“ a datum, kdy se daná editovaná podmínka uskuteční, „**Start date**“. Aplikace podporuje také hromadnou editaci. Hromadně lze editovat podmínky ze stejné množiny (se stejným jménem) a pouze položky se stejnými parametry. Možné jsou tedy: „**Active, Profile, Time change, Time change back**“. Pokud se uživatel pokouší editovat podmínky z různých množin, nebo dané podmínky nemají ani jeden stejný parametr, tak je uživatel upozorněn varovným hlášením (*obrázek 19*). Ty parametry, které nejsou stejné jsou při editaci skryty.

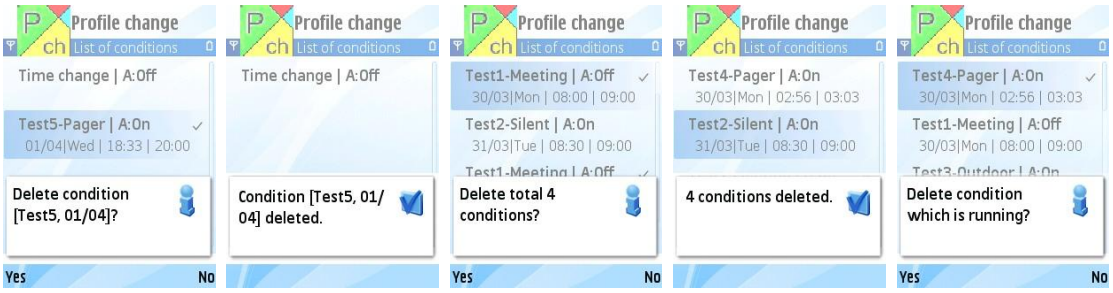


Obrázek 19: Editace 9 podmínek z množiny Test3, editace 2 podmínek z množiny Test2 pouze se stejným parametrem Profile, varovná hlášení

Při ukládání zeditovaných podmínek se samozřejmě kontrolují opět překrytí a další parametry. Editace právě probíhající podmínky nelze uskutečnit (*obrázek 19*). Pokud se během editace stane podmínka aktivní, nelze ji uložit. Zobrazuje se opět varovné hlášení. Editaci lze pouze opustit volbou „**Back**“.

4. Mazání podmínek

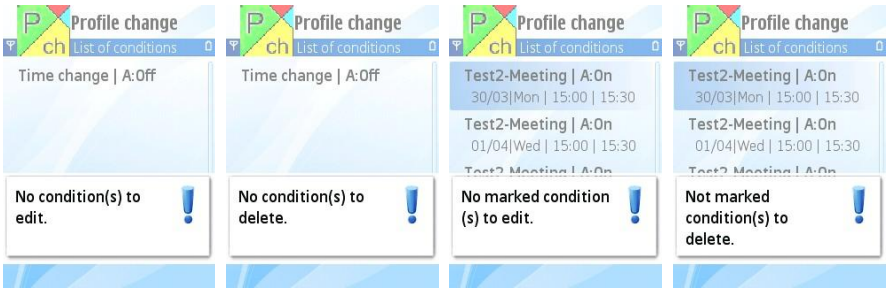
Mazat lze jednu nebo více podmínek z určité množiny, nebo z více z různých množin najednou (*obrázek 20*). Při mazání aktuálně probíhající podmínky je na to uživatel upozorněn. Pokud ji i přesto chce smazat, **změní se na profil, ze kterého měnila, ihned po vymazání**. Pokud uživatel maže více podmínek a jedna z nich je aktivní, je možno zvolit, zda ji chce také smazat či ne.



Obrázek 20: Mazání podmínek

5. Označování podmínek

Při editaci a mazání musí být jedna či více položek označeny. V opačném případě se objevují varovná hlášení. Stejně tak při pokusu mazat či editovat při neuložené žádné podmínce (*obrázek 21*).



Obrázek 21: Varovná hlášení při pokusu editovat či mazat při prázdným seznamu či neoznačené ani jedné podmínce

Položky lze označovat pomocí menu „**`Mark/UnMark`**“ nebo také pomocí klávesových zkratek, které lze vidět na *obrázku 14* u těchto položek. Položky „**`Mark`**“ a „**`UnMark`**“ označují a odznačují vždy jednu, která je aktuálně zvýrazněná. Toto lze provést pomocí středového tlačítka na joysticku `[ok]`. Položky „**`MarkAll`**“ a „**`UnMarkAll`**“ (klávesová zkratka `[*]` a `[#]`) označují a odznačují všechny uložené podmínky v seznamu. Položky „**`MarkAllSame`**“ a „**`UnMarkAllSame`**“ (klávesová zkratka `[7]` a `[9]`) označují a odznačují všechny uložené položky, které jsou ze stejné množiny (mají stejné jméno s aktuální položkou zvýrazněnou). Položky „**`MarkSameDay`**“ a „**`UnMarkSameDay`**“ (klávesová zkratka `[8]` a `[0]`) označují a odznačují všechny položky, které jsou ze stejné množiny a navíc mají vygenerované startovní datum ve stejný den v týdnu. Položky lze také zvýrazňovat podržením tlačítka shift (tlačítko tužka) a tlačítek nahoru a dolů na joysticku.

6. Změna na dobu

Pro události, které se vyskytnou náhle, slouží nastavení „**`Time change`**“ (změna na určitý čas). Nastavení indikuje první položka v seznamu podmínek „**`List of conditions`**“ ihned po spuštění aplikace („**`Time change | A:On`**“, „**`Time change | A:Off`**”).

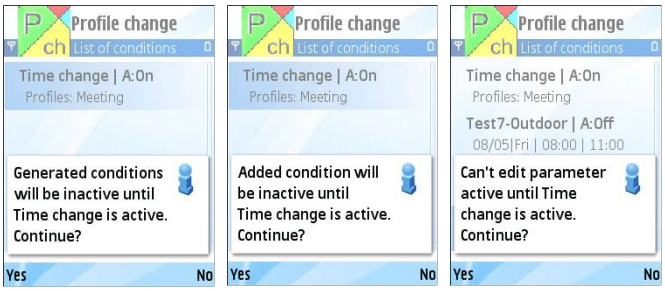
Obrázek 22: Nastavení změny na dobu

Nastaví se profil a jeho doba trvání. Takto lze postupně nastavit 1 až 3 profily, které se postupně po sobě budou měnit. V menu stiskem tlačítka „**`Next change`**“ se definuje další profil, nebo stiskem tlačítka „**`Set`**“ se nastavení ukončí a uloží (*obrázek 22*). U tohoto nastavení je kontrolováno pouze, zda nějaká doba trvání byla zadána (*obrázek 23*). Po ukončení nastavení se ihned změní na první definovaný profil a začne se počítat nastavená doba jeho trvání. Pokud uživatel nastavil změn více (maximálně 3), tak po vypršení nastavené doby se změní na další nastavený atd.

Po ukončení posledního nastaveného profilu se změní na původní profil před všemi změnami. Nastavení změna na čas zneaktivní všechny aktivní podmínky (*obrázek 23*).

Obrázek 23: Aktivní změna na dobu s třemi postupnými změnami, náhled, při zrušení změny na dobu, varovné hlášení

Při stisku středového tlačítka joysticku na změnu na čas se provede náhled jednotlivých nastavených změn (*obrázek 23*). Zrušit změnu na dobu lze pomocí menu nebo klávesové zkratky `[6]`. **Při zrušení se změní na profil před změnou** a provede se opět zaktivnění těch podmínek, které byly před započetím změny aktivní. Stejně jak při řádném ukončení. Poté se mění profily již podle definovaných podmínek. Při rušení změny na čas, která není aktivní, se zobrazí varovné hlášení (*obrázek 23*). Když je aktivní nastavení změna na čas, tak se při generování a přidání podmínek dané podmínky po přidání chovají jako neaktivní společně s ostatními. Dále při aktivním nastavení změna na čas nelze u podmínek editovat aktivitu. Uživatel je na tyto skutečnosti upozorněn varovným hlášením (*obrázek 24*).



Obrázek 24: Varovná hlášení při aktivním nastavení změna na čas

Příloha 2 – Postup jak aplikaci odsimulovat

Stejné informace lze najít v souboru README na přiloženém CD.

1) Stažení a instalace nástroje `Carbide.c++ 2.0` verze Developer:

- nástroj je zdarma ke stažení po registraci na http://www.forum.nokia.com,
- jako požadavek pro instalaci je uveden **`ActiveState ActivePerl`**, aplikace byla překládána pod verzí **`ActivePerl-5.10.0.1004`**,
- vždy při otevření nástroje se volí **`workspace`** - pracovní složka, kde jsou umístěny projekty,
- projekt do této složky umístíme.

2) Stažení a instalace vývojového balíčku SDK:

- v projektu je použit balíček: **`S60 3rd Edition FP1 (Symbian OS v9.2)`**,
- tento balíček lze najít na adrese: http://www.forum.nokia.com,
- zde na stránce je třeba zvolit verzi: **`3rd Edition, FP1 (355 MB)`**.

3) Načtení projektu do vývojového nástroje `Carbide.c++`:

- každá aplikace (projekt) obsahuje soubor **`bld.inf`**,
- v okně **`Project Explorer`** po stisku pravého tlačítka myši se nachází položka **`Import`**,
- položka `Import` se nachází také: **`File -> Import`**,
- po otevření nabídky **`Import`** zvolíme zdroj importu: **`Symbian OS -> Symbian OS Bld.inf file`**,
- najdeme soubor `Bld.inf` projektu, ten se nachází ve složce `group`,
- objeví se, pod jakou verzí OS budeme chtít projekt kompilovat,
- zvolíme instalovanou **`S60_3rd_FP1`**,
- v **`MMP Selection`** necháme zaškrtnuté **`Icons_aif_scalable_dc.mk`** a **`PCH_v110.mmp`**, odškrtnuté **`Exclude extension makefiles`** a **`Exclude test components`**,
- v **`Project Properties:`** Project name ponecháme **`PCH_v110`**,
- tlačítko **`Finish`**.

4) Kompilace a spuštění v emulátoru:

- před kompilací je **nutné** stiskem pravým tlačítkem myši na složku projektu v Project Explorer zvolit: **`Build Configurations -> Set Active -> Emulator Debug (WINSCW) [S60_3rd_FP1]`**,

Obrázek 25: Úspěšně spuštěný emulátor, označeno tlačítko menu

- kompilovaný program se poté nachází ve složce **`Installed`**,
- tlačítko `tužka` se na emulátoru emuluje pomocí tlačítka `shift` na klávesnici, toto tlačítko slouží například při označování položek u seznamů typu `Markable` (podrží se a stlačuje se tlačítko nahoru či dolů).
- Aplikace uložené informace ukládá do souboru `Conf_condition.dat`. Ten se nachází v systému Symbian v `C:\System\Apps\PCH_v110\Conf_condition.dat`. Při ponechání defaultních cest při instalaci vývojového balíčku SDK, tak ho lze najít v emulátoru `C:\Symbian\9.2\S60_3rd_FP1\Epoc32\winscw\c\system\Apps\PCH_v110\Conf_condition.dat`.

5) Vytvoření nepodepsaného instalačního souboru .sis:

- stiskem pravým tlačítkem myši na složku projektu v Project Explorer zvolit: **`Build Configurations -> Set Active -> Phone Release (GCCE) [S60_3rd_FP1]`**,
- stiskem pravým tlačítkem myši na složku projektu zvolit: **`Properties`**,
- zde v nastavení najdeme **`Carbide.c++ -> Build Configurations`**,
- v záložce **`SIS Builder`** zvolíme **`Add`** a objeví se okno **`SIS Properties`**,
- **`PKG File`** zvolíme **`PCH_v110.pkg`**,
- zvolíme také v Signing Options: **`Don't sign sis file`**,

- poté pravým tlačítkem na složku projektu v Project zvolit: **`Build Project`**,
- ve složce **`sis`** se objeví nepodepsaný instalační soubor sis,
- nepodepsaný instalační soubor je třeba před instalací podepsat.

6) Podepsání instalačního souboru pro testovací účely:

- to lze pomocí *Open Signed Online*,
- odkaz: https://www.symbiansigned.com,
- tam je třeba zvolit **`IMEI` číslo telefonu** (to lze zjistit zadáním kombinace `*#06#`),
- potvrzovací email,
- nutné je zde pro správnou funkci aplikace zaškrtnout Capability: **`WriteDeviceData`**, toto je nutné pro správnou funkčnost všech funkcí `Profile Engine Wrapper API`,
- nejprve dojde potvrzovací email, poté na email dojde odkaz, odkud je možné podepsanou aplikaci stáhnout.

Poznámka:

Aplikace byla testována na telefonu **N95 (S60 3rd FP1, Symbian 9.2)**. Telefony, které běží na platformě **S60 3rd FP1, Symbian 9.2**: Nokia N95, Nokia N82, Nokia N77, Samsung SGH-i450, Samsung SGH-i550, Samsung SGH-i520, Samsung SGH-i560, Samsung G81, LG JOY

Příloha 3. Všechny soubory aplikace

Popis jednotlivých zdrojových souborů se nachází v *kapitole 4.3*.
Ve složce `Program_documentation` se nachází programová dokumentace k aplikaci ve formátu `html`. Programová dokumentace byla vygenerována pomocí nástroje `Doxygen`.