České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídicí techniky

Vizualizační prostředí pro domovní automatizaci

Diplomová práce

Bc. Ivo Kubita

Vedoucí práce:
Ing. Pavel Burget Ph.D.

Praha 2009

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Ivo Kubita**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Vizualizační prostředí pro domovní automatizaci**

Pokyny pro vypracování:

Cílem práce je navrhnout a realizovat vizualizační prostředí pro sledování a konfiguraci automatizované budovy. Práce je zaměřena především na malé budovy jako například rodinné domy.

1. Seznamte se s existujícím systémem pro vizualizaci a konfiguraci automatizovaného domu, založeným na komponentech DAMIC.
2. Proveďte analýzu požadavků systému DAMIC vyvíjeného na katedře řídicí techniky. Zpracujte detailní návrh architektury vizualizačního prostředí tak, aby splňoval požadavky tohoto systému. Architektura musí být modulární, aby bylo možné později systém rozšířit o další funkční prvky.
3. V .NET realizujte vizualizační prostředí pro systém DAMIC jako samostatnou aplikaci. Aplikaci vytvořte tak, aby bylo možné ji použít i v distribuovaném prostředí Internetu.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Burget

Platnost zadání: do konce zimního semestru 2009/2010

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

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

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 20. 4. 2009

Prohlášení

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

V Praze, dne: 22.5.2009 

podpis

Poděkování

Děkuji panu Ing. Pavlu Burgetovi Ph.D. za vedení a pomoc, kterou mi poskytl při zpracování tématu mé diplomové práce.

Rovněž bych rád poděkoval kolegům Ing. Ondřeji Nývltovi a Ondřeji Fialovi za spolupráci, konzultace a rady, které pomohly vylepšit a rozvinout nejen tuto práci, ale i celý projekt domovní automatizace.

ANOTACE

Klíčová slova: domovní automatizace, .Net Remoting, WPF, Modbus, vizualizace, uLan.

Diplomová práce se zabývá vytvořením univerzálního konfiguračního a vizualizačního softwaru pro domovní automatizaci. Je zde rozebrána architektura systému, metody vzdáleného přístupu a meziprocesní komunikace. V rámci práce byl vytvořen model domovního automatizačního systému a aplikace pro konfiguraci a vizualizaci systému Damic.

ANNOTATION

Key words: home automation, .Net Remoting, WPF, Modbus, visualisation, uLan.

The thesis deals with the creation of universal configuration and visualisation software for home automation. Topics such as architecture of the system, remote access methods, and interprocess communications are analysed here. A model of a home automation system, as well as an application for the configuration and visualisation of the system Damic were designed as a part of the work.

OBSAH

1 ÚVOD

Automatizační technika se začíná pomalu dostávat do sféry domácností. Různé firmy nabízejí řadu systémů, které se snaží učinit bydlení příjemnějším, komfortnějším a ekonomičtějším. Firmy používají rozdílné technologie pro propojení jednotlivých zařízení v systému, od bezdrátových distribuovaných systémů až po centralizované systémy. Domovní automatizací se rozumí především ekonomické a komfortní řízení vytápění, ovládání světel, ventilace a žaluzií. Některé z těchto systémů přenášejí po společných rozvodech domovní automatizace i video, zvuk, případně data, jiné připojují zabezpečení domu nebo přídavné periferie pro automatické čištění bazénu a podobně.

Každý z těchto systémů je potřeba vhodným způsobem vizualizovat, konfigurovat a řídit. Kdo by si v dnešní době nepřál ovládat svůj dům mobilním telefonem nebo vše pohodlně nastavit na svém notebooku? Cílem této práce je navrhnout a vytvořit takovou architekturu vizualizačního a konfiguračního softwaru, který bude jednoduchý pro uživatele, avšak bude dostatečně univerzální, aby bylo možné jej s minimálním úsilím použít i pro jiný typ domovní automatizace.

V úvodních kapitolách se práce zabývá analýzou požadavků systémů domovní automatizace na vizualizační a konfigurační prostředí. Jednotlivé požadavky jsou popsány z pohledu uživatelů systému a shrnuty v konkrétní soubor nároků na aplikaci a její části. Tyto nároky jsou dále použity pro návrh architektury. V následujících kapitolách je rozebrána navržená architektura, její jednotlivé části a přehled použitých technologií.

Poslední kapitola poskytuje příklad implementace architektury a využití technologií, které jsou popsány v kapitolách předchozích. Zatímco předchozí kapitoly jsou spíše koncepčního charakteru, kapitola 11 obsahuje popis konkrétní realizace jednotlivých koncepčních prvků a doplňuje tak vlastnosti a postupy vysvětlené dříve. Zdrojový kód realizace je obsažen na přiloženém CD.

2 POŽADAVKY NA SYSTÉM

Hlavním kritériem pro aplikaci je určení jejího použití. Aplikace je určená jak pro koncového uživatele, tak pro instalační firmu.

a) Pro koncového uživatele - obyvatele rodinného domu
b) Pro koncového uživatele - správce menšího rekreačního objektu
c) Pro firmu, která provádí instalace systémů domovní automatizace
d) Pro firmu, která bude řešit technické problémy nebo konfiguraci na dálku

Koncový uživatel

Typický koncový uživatel požaduje od systému domovní automatizace především jednoduché a intuitivní ovládání, ale pokud je to možné, i úsporu energie. Bez znalostí automatizačního systému (dále systém) může zobrazit jednotlivé stavy systému, jako například teploty v místnostech, rozsvícená světla, otevřená okna, spuštěné ventilátory atd. Tyto stavy bude schopen jednoduše ovládat – kliknutím na komponentu světla v systému se světlo rozsvítí i v domě. Všechny tyto funkce může uživatel provádět na dálku. Pro jednotlivé místnosti může nastavit předdefinované teplotní programy nebo může definovat vlastní program. Pokud je některá místnost delší dobu neobsazená, uživatel nastaví program pro minimální vytápění. Teplotní programy lze aktivovat nebo deaktivovat v definovaný časový okamžik – například pokud uživatel jede na dovolenou, nastaví datum a čas aktivace minimálního vytápění pro celý dům a datum a čas pro deaktivaci tohoto programu. Stejné funkčnosti využije i správce rekreačního zařízení, který může jednoduše na dálku zapínat nebo vypínat regulaci vytápění v pokoji nebo v apartmánu. V možnostech uživatele je i dočasná změna teplotního režimu na konstantní teplotu. Doba i teplota se dají nastavit. Technicky pokročilý uživatel může mít přístup ke změnám konfigurace systému, například k vazbám mezi jednotlivými zařízeními nebo k detailnímu nastavení zařízení, podobně jako technik instalační firmy.

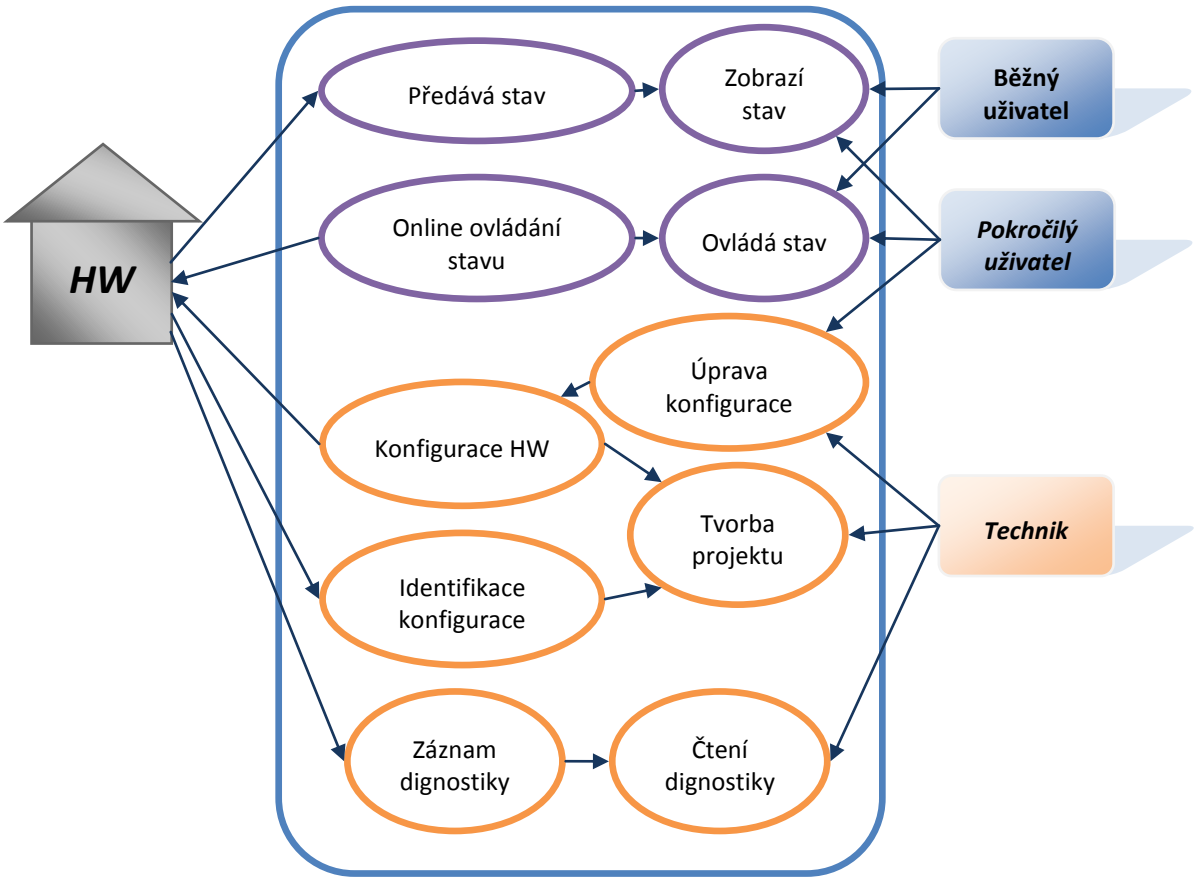
Technik

Po instalaci hardwaru je nutné systém správně identifikovat a provést počáteční konfiguraci. Technik zjistí počet a typ jednotlivých zařízení v systému. Ke každému zařízení přiřadí popis s jeho fyzickou polohou – identifikuje fyzické zařízení v seznamu aplikace. Poté nastaví počáteční konfiguraci jednotlivých zařízení, rozdělí je podle místností a nastaví vazby mezi zařízeními – například tlačítku vypínače

přiřadí akci určitého světla. Pro vizualizaci nahraje do aplikace půdorys (případně jiný náhled) domu a rozmístí zde vizualizační komponenty. Pro ověření korektní funkčnosti konfigurace může z vizualizace simulovat jednotlivé akce uživatelů.

Pokud dojde k nějaké chybě v systému nebo pokud uživatel bude potřebovat zásah technika, stačí kontaktovat firmu; technik se připojí k systému na dálku a uživateli chybu opraví nebo doporučí vhodné řešení. Technik je rovněž schopen systém nastavit ze svého pracoviště nebo z kteréhokoli jiného místa. Systém umožňuje pomocí diagnostiky varovat uživatele nebo přímo instalační firmu o případných problémech. Pokud má například některé zařízení poruchu, může o tom být informován jak uživatel, tak přímo firma. Firma může se svolením uživatelů shromažďovat diagnostické údaje ze systému a využít jich pro zlepšení systému.

Obrázek 2.1 charakterizuje přístup uživatelů k jednotlivým funkcím.



Obrázek 2.1: Funkce systému domovní automatizace

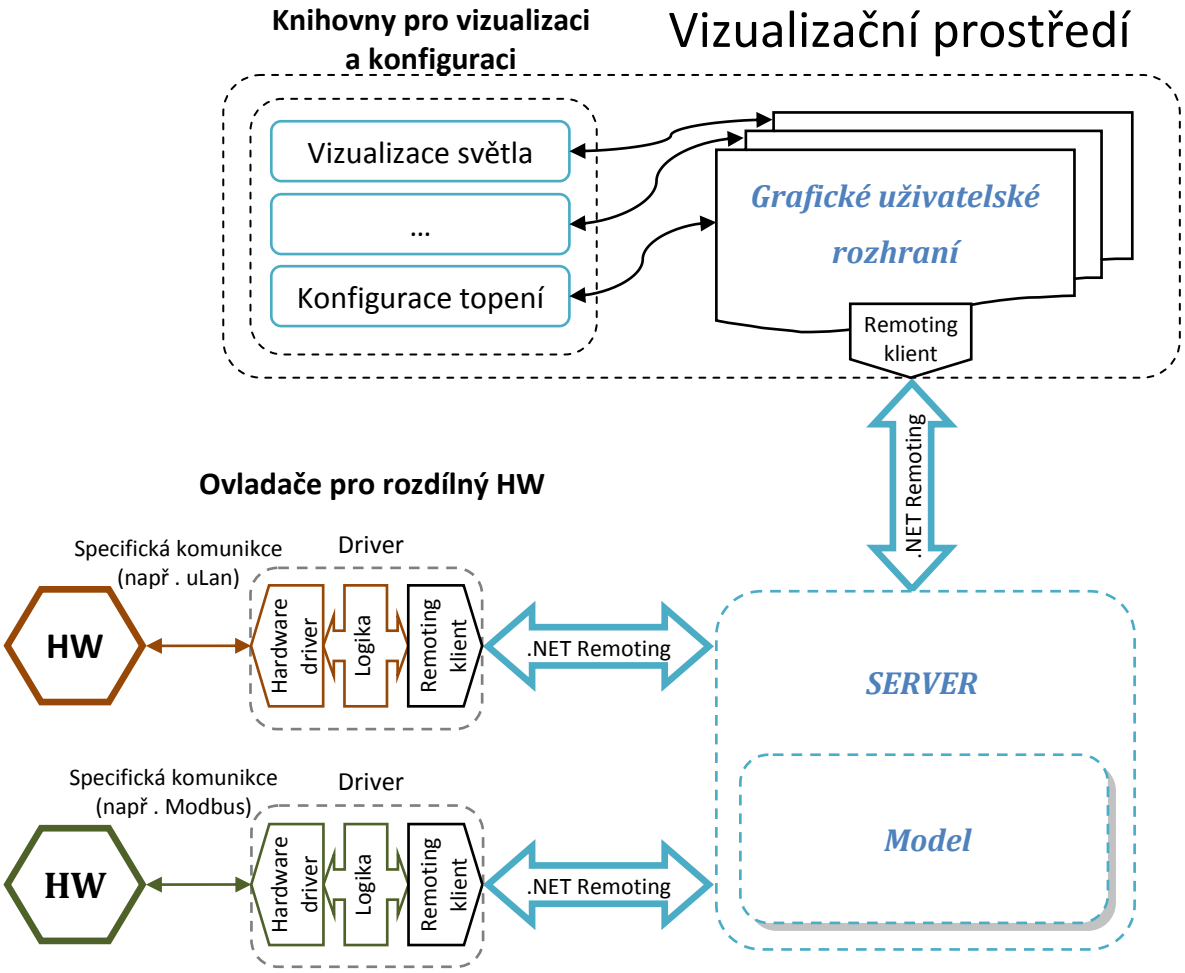
Z těchto scénářů vyplývají následující požadavky, které by měla architektura systému umožňovat:

a) Vizualizace stavů jednotlivých částí systému – světla, teploty, ventilátory, topení, kotel, okenní kontakty, atd.

b) Konfigurace všech částí systému – nastavení světel, regulátorů, topných křivek, atd.
c) Centrální správa jedné a více budov – technik i uživatel jsou schopni konfigurovat či zobrazovat stavy systému vzdáleně.
d) Priority uživatelů – pokud dva uživatelé budou nastavovat stejnou část systému, bude změna povolena pouze uživateli s vyšší prioritou.
e) Více typů hardwaru – systém bude umět pracovat s větším počtem rozdílných platforem souběžně a kombinovat je v projektu.
f) Zaznamenání historie – zaznamenání teplot v pokojích v průběhu roku, atd.
g) Alarmy – bezpečnostní nebo událostní – pokud teplota v kotli překročí mez, bude o tom uživatel informován varovnou zprávou.
h) Dynamická konfigurace – konfigurace může záviset na vnějších podmínkách. Systém může kontrolovat předpověď počasí a automaticky nastavit topné plány podle vnějších vlivů.
i) Různé typy vizualizací a ovládání – lokální, webová, sms, mail, atd.
j) Diagnostika systému – software umožňuje zaznamenání a rozbor všech zpráv, které se objeví na společné sběrnici

3 ARCHITEKTURA SYSTÉMU

Hlavním požadavkem bylo, aby systém byl přístupný na dálku. Proto byla základní koncepce zvolena aplikace typu klient – server. Na serveru je umístěn model, který je poskytován všem klientům. Server neumí pracovat s konkrétními daty v modelu, pouze s modelem jako celkem, ale poskytuje klientům funkce pro práci s konkrétními částmi modelu. Pokud některý klient změní některou součást modelu, server je schopen o této události uvědomit ostatní klienty. Použitá technologie pro komunikaci klientů se serverem je .NET remoting se sdíleným rozhraním – všichni klienti přistupují na server pomocí definovaného rozhraní.



Obrázek 3.1: Schéma částí vizualizačního software

Klient je jakákoli aplikace, která se připojuje k serveru přes remotingové rozhraní - grafické rozhraní pro uživatele, stejně jako driver pro určitý hardware.

Klient, který přistupuje přímo k hardwaru, se nazývá driver. Driver rozlišuje proměnné na stavové a konfigurační. Konfigurační proměnné popisují nastavení systému, například vazby mezi jednotlivými částmi, teplotní programy, atd. Stavové proměnné určují aktuální stav systému, například rozsvícené světlo, zapnuté topení, teplotu v místnosti, atd. Každý driver je schopen stáhnout kompletní konfiguraci a stavy z hardwaru, vytvořit z konfigurace model a ten posléze nahrát na server. V případě, že se změní stav jakéhokoli zařízení v systému, driver obnoví příslušné proměnné v modelu na aktuální. Pokud některý jiný klient změní stavové proměnné, driver je nahraje do příslušných zařízení. Detailní popis přenosu zpráv je uveden v kapitole 11. Uživatel může změnit kompletní konfiguraci systému, kterou driver nahraje po potvrzení uživatelem.

Klient, který má přístup pouze do modelu a vyžaduje interakci uživatele, se nazývá uživatelské rozhraní. Uživatelské rozhraní umožňuje předat uživateli informace o stavových proměnných, měnit je a měnit konfiguraci systému. Uživatelské rozhraní umí pracovat s modelem jako s celkem, pro práci s jednotlivými částmi využívá přídavných knihoven. Každá z těchto knihoven umí pracovat s určitým typem komponenty pro konkrétní hardware. Pokud může uživatel ovládat software pomocí interaktivních grafických prvků, jedná se o grafické uživatelské rozhraní (GUI). V GUI je pro zvolené komponenty dostupná knihovna, která umí stavy a konfiguraci vhodně vizualizovat. Knihovny jsou schopny přijímat zprávy o změně stavu příslušných komponent a reagovat na tyto události – například driver zjistí změnu stavu světla, zapíše ji do modelu, server na tuto událost zareaguje rozesláním příslušné zprávy a v klientovi tuto zprávu zachytí příslušná knihovna, která změní grafickou reprezentaci dané komponenty (např. obarví ikonu světla). GUI umí model stáhnout ze serveru a lokálně uložit na disk uživatelova PC, případně zaslat serveru zprávu s požadavkem na uložení modelu na server. GUI umožňuje lokálně uložený model nahrát na server nebo stáhnout ze serveru seznam všech uložených modelů a vybrat jeden pro nahrání. Uživatel takto může mít několik konfigurací, kterými může nastavit celý dům. K těmto konfiguracím lze přistupovat vzdáleně, pokud jsou uložené na serveru. Uložený model obsahuje navíc oproti hardware informaci o fyzické poloze jednotlivých zařízení, jejich rozmístění do místností, případně jména zařízení a komentáře.

4 MODEL

Konfiguraci a strukturu systému je potřeba vhodně udržovat v paměti za běhu systému, zprostředkovat k ní jednoduchý přístup a uložit ve formátu použitelném jiným softwarem. Model musí být dostatečně univerzální, aby jím bylo možné reprezentovat jakýkoli systém domovní automatizace. Na základě těchto požadavků je nutno vytvořit pro celý systém model, který by všechny tyto nároky splňoval.

Požadavky na strukturu a obsah modelu:

- Obecné informace o souboru a modelu
- Popis a nastavení ovladače, kterým se přistupuje k hardwaru
- Typy jednotlivých zařízení obsažených v modelu
- Seznam všech zařízení obsažených v modelu
- Konfigurace a stavy všech zařízení v modelu
- Rozlišení stavových a konfiguračních proměnných
- Obecné nastavení pro grafické rozhraní
- Nastavení všech zařízení pro grafické rozhraní
- Vlastnosti pro grafické reprezentace jednotlivých komponent
- Filtrační kategorie – například místnosti
- Popis a vlastnosti vztahů mezi jednotlivými komponentami

Při návrhu modelu bylo použito několik vlastností z PlcOpen XML schématu, viz (1). Tento model se zprvu zdál použitelný i pro model domovní automatizace, nicméně při implementaci se projevila velmi těsná orientace na PLC a těsnější vazba na programování, než je potřeba pro domovní automatizaci. Proto byl tento model použit pouze jako inspirace pro vytvoření modelu nového. Podle schématu PlcOpen je možné definovat jakýkoli datový typ pro jakoukoli proměnnou, popsat program systému a ten rozčlenit do jednotlivých úkolů. Tyto vlastnosti jsou nutné pro popis programu PLC nicméně pro systém domovní automatizace nejsou potřeba. Možnost tvorby vlastních datových typů pro proměnné také zvyšuje náročnost zpracování modelu. Tyto vlastnosti byly zcela vypuštěny.

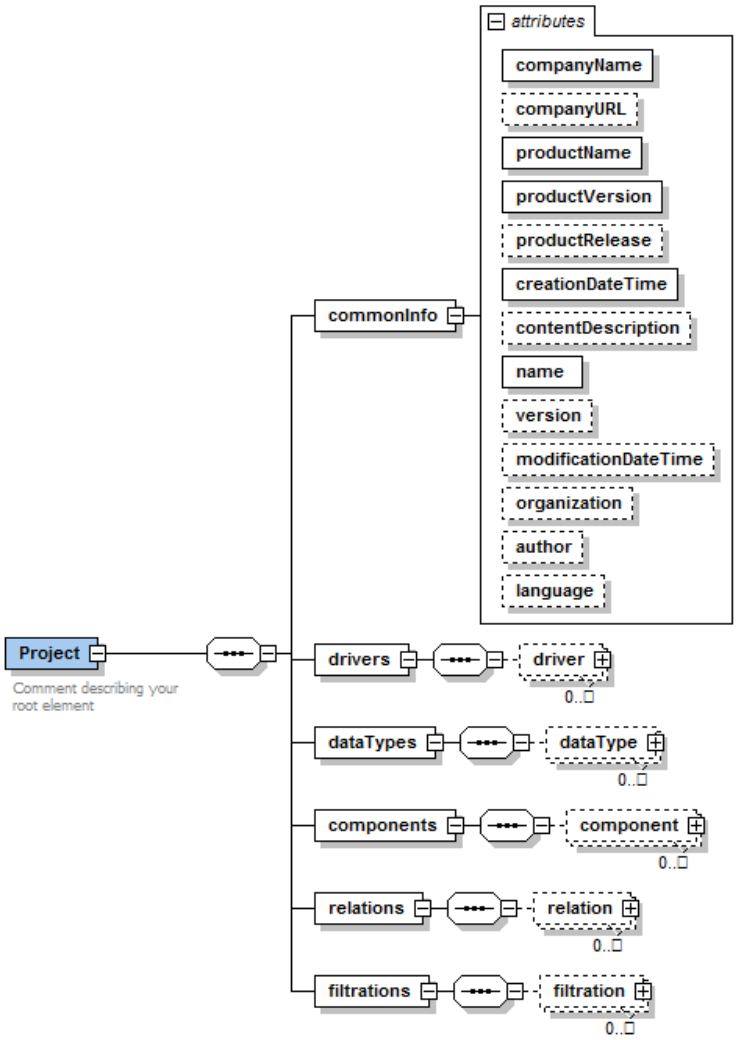
4.1 Navržený model

Model byl navržen jako XML schéma. Tento formát je velmi univerzální, lze s ním pracovat v podstatě na jakémkoli systému a existuje velké množství nástrojů, které jej podporují.

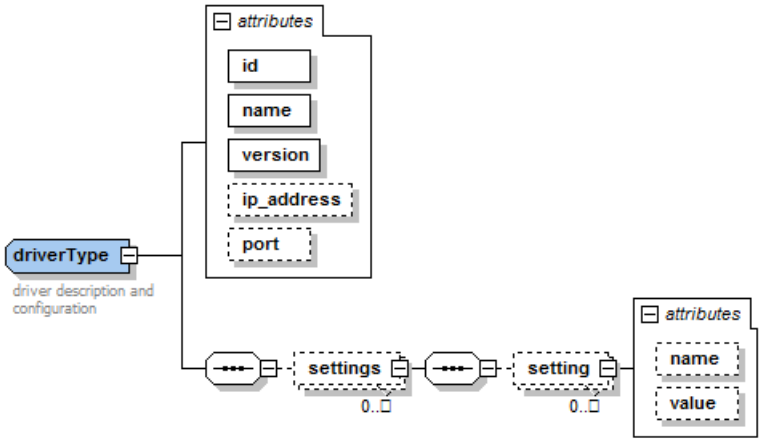
Sekce „*commonInfo*“ popisuje obecné nastavení a vlastnosti projektu (viz. Obrázek 4.1). Zde jsou uloženy informace o firmě, která soubor s modelem vytvořila („*companyName*“, „*companyURL*“), jméno a verze produktu („*productName*“, „*productVersion*“, „*productRelease*“), datum vytvoření a změny („*creationDate*“, „*modificationDateTime*“), popis obsahu („*contentDescription*“), jméno a verze modelu („*name*“, „*version*“), organizace, která model používá („*organisation*“), autor, který model vytvořil nebo změnil, a jazyk, ve kterém jsou napsány popisy zařízení. V sekci „*dataTypes*“ v Projektu jsou uloženy všechny základní typy komponent použité v projektu. Z nich se vytvoří nová komponenta vstupující do sekce „*components*“, kde už má vazbu na konkrétní fyzickou komponentu. Sekce „*drivers*“ obsahuje seznam všech hardwarových platforem, které jsou použity v projektu. Vztahy mezi jednotlivými komponentami mohou být popsány přímo ve vlastnostech komponent nebo explicitně v sekci relations. Filtrační kategorie „*filtrations*“ slouží například pro rozdělení komponent do místností.

4.2 Definice ovladače

Obrázek 4.2 popisuje definici typu ovladače – „*driverType*“ popisujícího položky „*driver*“. Projekt obsahuje seznam všech použitých ovladačů, neboť předpokládáme, že v budoucnu by měl software umět pracovat s větším množstvím hardwarových systémů současně. Každý ovladač má povinné atributy id, jméno a verzi. Id musí být v daném projektu unikátní pro každý ovladač. Jméno definuje použitou hardwarovou platformu. Toto jméno určuje, která část systému patří pro který konkrétní hardware. Atribut verze definuje, pro kterou verzi hardwarového ovladače je konfigurace navržená. Atributy ip adresa a port jsou volitelné a využívají se pouze v případě, že je hardware dostupný přes tcp/ip protocol. Seznam dalších nastavení „*settings*“ umožňuje definovat jakékoli doplňkové parametry či nastavení driveru.



Obrázek 4.1: Popis základních prvků modelu



Obrázek 4.2: Popis definice driveru

4.3 Definice typu komponenty

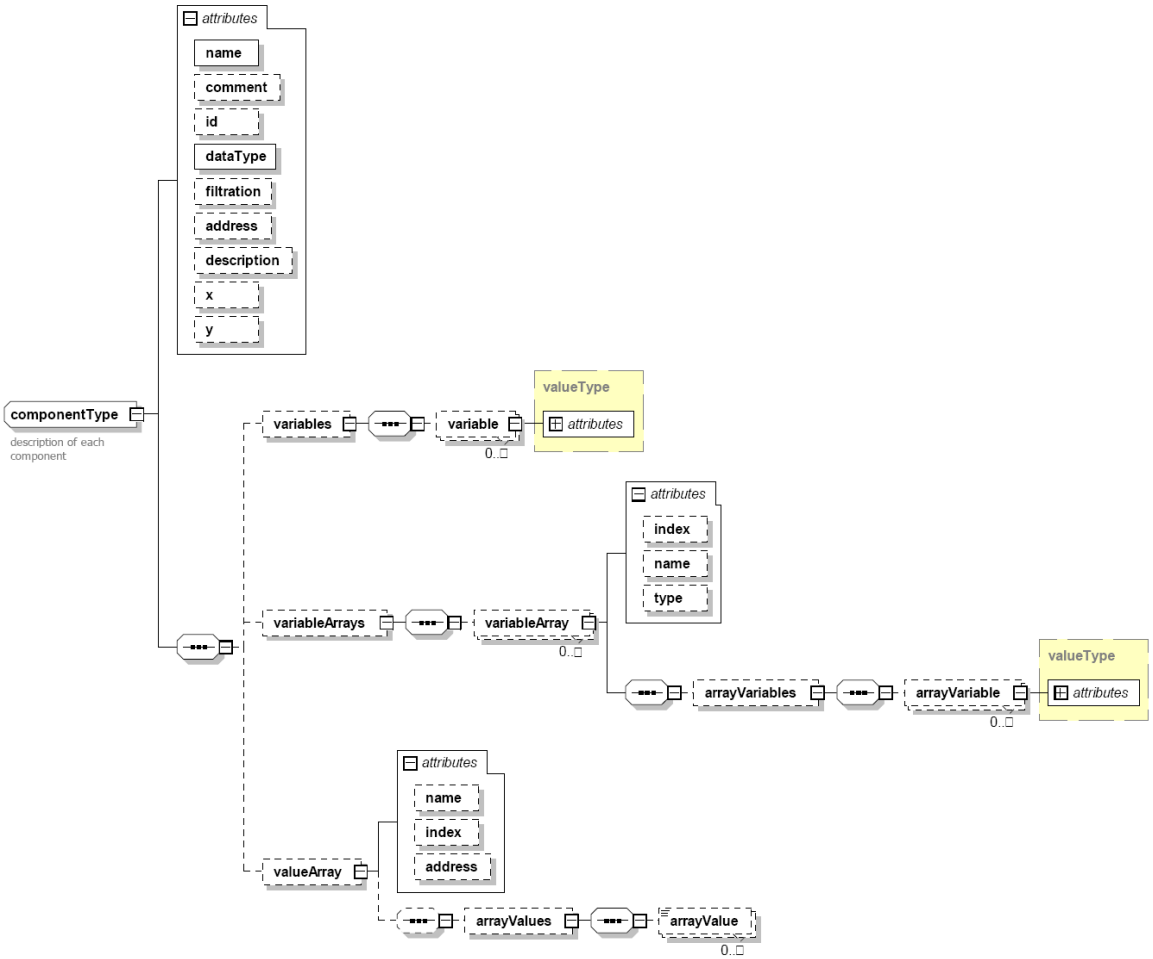
Komponenta systému je jakákoli část systému, která se dá samostatně popsat, například vypínač, akční člen, termostat, atd. Termín „*componentType*“ je použit jak pro definici různých typů použitých komponent, tak i pro definici každé jednotlivé komponenty.

Obrázek 4.3 zachycuje grafické znázornění komponenty systému. Vytvořit popis jakékoli komponenty, která se v systému může objevit, je poměrně složité; model by se tudíž mohl stát velmi komplikovaným.

Proti jisté univerzálnosti stojí fakt, že model musí být přístupný online. Pokud by byl příliš obecný, byl by mnohem rozsáhlejší, a to by mohlo vést k problémům s rychlostí čtení a zápisu do modelu. Navržená obecnější verze byla velmi podobná XML schématu PlcOpen, nicméně práce s tímto modelem byla velmi pomalá. U komponent domovní automatizace se nepředpokládá velká složitost, a proto byl popis komponenty navržen jednodušší a omezenější.

Atributy komponenty datový typ a jméno jsou povinné a slouží k identifikaci každé komponenty. Datový typ udává základní datový typ v sekci „*dataTypes*“. Následující atributy jsou komentář, id, filtrační kategorie, adresa, popis a poloha v grafickém rozhraní (x a y).

Zbývající vlastnosti komponenty jsou popsány ve třech polích. Právě tady se projevilo zjednodušení modelu. Sekce „*variables*" je pole proměnných typu „*valueType*". V této sekci mohou být proměnné společné pro celou komponentu. Sekce „*variableArrays*" je dvourozměrné pole proměnných typu „*valueType*". Zde mohou být například

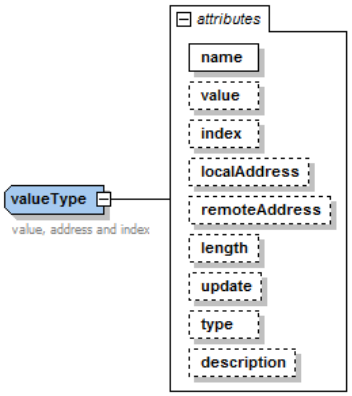


Obrázek 4.3: Popis definice komponenty

jednotlivé kanály v jedné komponentě (nastavení jednotlivých světel na stmívači), pokud jich komponenta obsahuje více. Každému kanálu je možné přiřadit index, jméno a typ. Poslední sekce „*valueArray*" je jednoduché pole proměnných typu „*string*".

4.4 Definice typu hodnoty

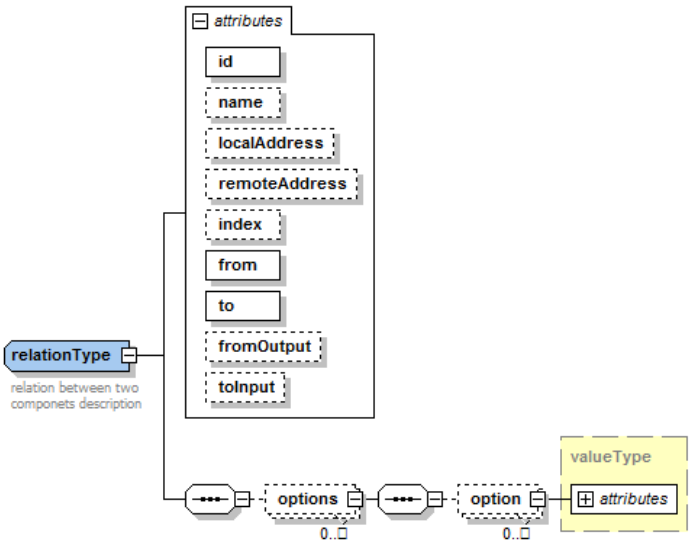
Obrázek 4.4 popisuje definici proměnné („*valueType*“). Tato proměnná je přiřazena konkrétní proměnné nebo veličině v zařízení. Povinným atributem je pouze jméno. Atribut „*value*“ obsahuje aktuální hodnotu. Tomuto atributu má význam přiřadit volání události „*OnChange*“ a sledovat ji, pokud se jedná o stavovou veličinu. Atributy index, lokální adresa a vzdálená adresa dohromady určují konkrétní umístění dané proměnné. Délka má význam, pokud je hodnota typu string, apod. Atribut „update“ definuje, zda je uvedena hodnota pro čtení (*r*) a zápis (*w*) a zda se má realizovat cyklické čtení (*ru*), případně zapisování (*wu*). V závorkách jsou uvedeny hodnoty atributu pro daný význam. Tyto hodnoty lze kombinovat, například hodnotu lze zapsat a zároveň cyklicky číst – „*ruw*“. Atribut „type“ udává, jakého typu je hodnota „*value*“. Typ je závislý na zařízení a není explicitně definován.



Obrázek 4.4: Popis definice hodnoty

4.5 Definice typu relace

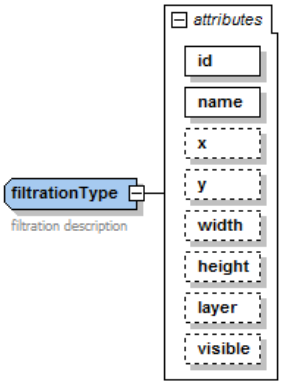
Relace udává vztah mezi dvěma komponentami. Pokud má komponenta vztah s více komponentami, jsou zde uvedeny všechny vztahy. Obrázek 4.5 reprezentuje definici typu popisujícího relaci. Kromě atributů se shodným významem, jako tomu bylo u předchozích typů, jsou zde navíc atributy odkud („*from*“), kam („*to*“), z jakého výstupu („*fromOutput*“), na jaký vstup („*toInput*“). Hodnoty atributů „odkud“ a „kam“ jsou id příslušných komponent. Pokud má relace nějaká další nastavení, jsou uvedena v sekci „*Options*“.



Obrázek 4.5: Popis definice relace

4.6 Definice typu filtrační kategorie

Filtrační kategorie („*filtrationType*“) lze použít například k rozdělení jednotlivých komponent do místností. Hlavní význam mají pro grafické uživatelské rozhraní. Atributy „*x*“, „*y*“, šířka „*width*“, výška „*height*“, vrstva „*layer*“a viditelnost „*visible*“ slouží pro zobrazení příslušných komponent.



Obrázek 4.6: Popis definice filtrační kategorie

5 MEZIPROCESNÍ KOMUNIKACE

V prostředí internetu je potřeba komunikovat mezi jednotlivými částmi distribuované aplikace. Tyto části mohou být umístěny na různých počítačích kdekoli v internetové síti. V dnešní době se stala vzdálená komunikace samozřejmostí a je vyvinuto velké množství nástrojů usnadňujících vývoj aplikací a poskytujících značnou rychlost a modularitu.

Kapitola byla vypracována za použití zdrojů (2),(3),(4).

5.1 Přehled technologií komunikace distribuovaných procesů v prostředí Windows

5.1.1 DCOM

Technologie DCOM (Distributed Component Object Model) je rozšířením COM architektury, která umožňuje vývoj aplikace sestávající z komponent. DCOM přidává této technologii možnost komunikace mezi jednotlivými částmi na různých počítačích. DCOM průběžně zasílá všem klientům zprávy v definovaných časových okamžicích, a tím kontroluje jejich dostupnost. DCOM bylo postupně nahrazeno následujícími technologiemi.

5.1.2 WEB Services

Web Services (Síťové služby) poskytly první jednoduchý způsob, který vedl k propojení rozdílných platforem a částí aplikací napsaných v rozdílných programovacích jazycích. Síťové služby představují bezstavová volání metod vzdálených komponent přes HTTP protokol ve formátu XML.

Kódování zpráv do XML se je možné provést dvěma různými způsoby:

První způsob kódování XML-RPC (XML- Remote Procedure Call) by se dal popsat jako velmi zjednodušená verze protokolu SOAP. Je implementován v mnoha programovacích jazycích.

Druhý způsob kódování SOAP (Simple Object Access Protocol) představuje mnohem rozmanitější sadu služeb, která poskytuje nejen volání metod vzdálených komponent, ale také definici WSDL a UDDI. WSDL (Web Services Description Language) je jazyk sloužící pro popis rozhraní služeb a UDDI (Universal Description, Discovery and

Integration) poskytuje funkce pro vyhledávání webových služeb. Všechny tyto protokoly a specifikace jsou založeny na XML, což dovoluje použití na mnoha platformách.

5.1.3 .NET Remoting

.NET Remoting je flexibilní a rozšiřitelný framework, který maximálně usnadňuje implementaci distribuovaných technologií. Zahrnuje mnoho funkcí z webových služeb, jako například SOAP, ale nezávisí na nich. Umožňuje rozdílné mechanismy přenosu (implicitně HTTP, TCP), různé metody kódování (implicitně SOAP, binární) a metody zabezpečení. Rozhraní nemusí být explicitně definované, je možné je získat ze serveru nebo přímo z .NET sestavení.

.NET Remoting využívá k zakódování jednotlivých objektů serializaci neboli seřazení (serialization, marsahlling). Pokud je objekt definován jako serializovatelný, je možné vytvořit jeho reprezentaci pomocí jakéhokoli serializátoru – například XML reprezentace, binární reprezentace atd.

.NET Remoting je proprietální technologie společnosti Microsoft, není tudíž podporována na jiných platformách než Windows.

5.1.4 Windows Communication Foundation

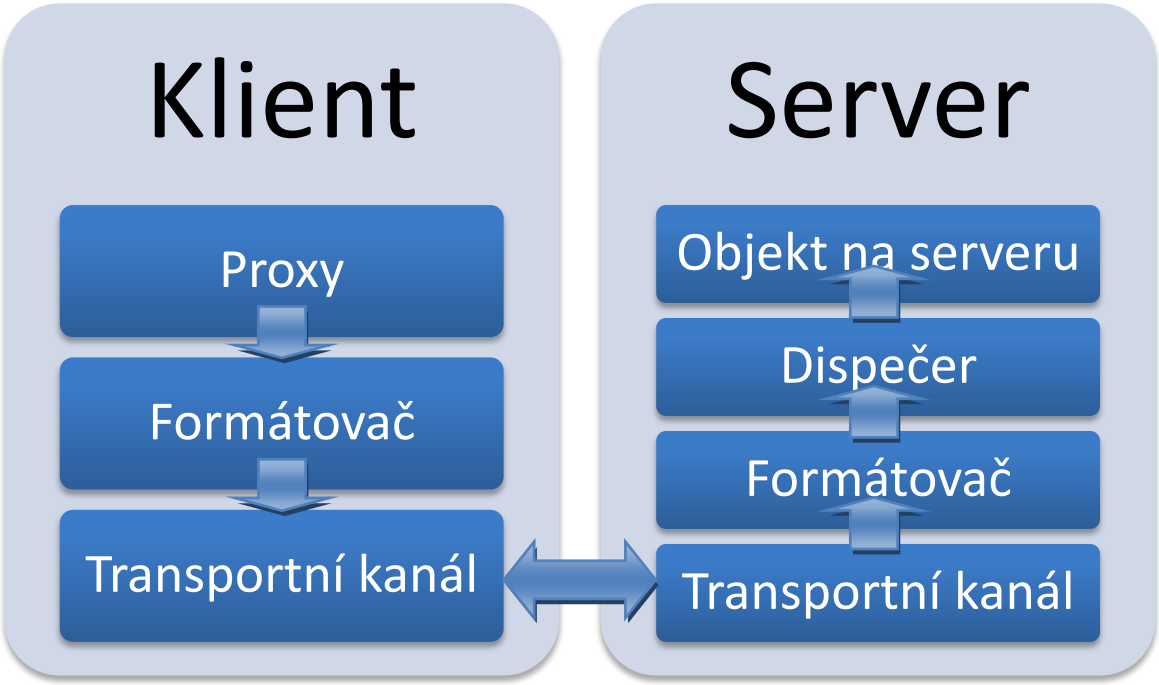
Windows Communication Foundation (WCF) vznikl jako reakce na velké množství komunikačních protokolů, nástrojů a frameworků. WCF ještě více zjednodušuje programování distribuovaných aplikací, neboť je vytvořen tak, aby uměl komunikovat se všemi technologiemi uvedenými v předchozích kapitolách. WCF je opět proprietální technologie společnosti Microsoft, nicméně tento framework umožňuje pomocí webových služeb komunikovat například s Java Enterprise Application.

5.2 .NET Remoting podrobněji

Kapitola vypracována za použití zdroje (2).

Komunikační technologií pro aplikaci domovní automatizace byl zvolen .NET Remoting. Výhodou této technologie je snadný přístup k objektu a velmi jednoduchá konfigurace, kterou je možné změnit bez zásahu do zdrojového kódu. Nevýhodou je složitější implementace asynchronních událostí, použitých pro předávání aktuálních

stavů jednotlivým klientům. V souvislosti s předáváním těchto událostí je také omezen maximální počet klientů na několik desítek, případně stovek, a to je pro domovní automatizaci zcela dostatečné. Technologie nabízí vývojovým pracovníkům a administrátorům širší nabídku protokolů a formátů než jakýkoli předchozí komunikační mechanismus. Obrázek 5.1 představuje zjednodušený popis architektury .NET remoting. Pokud má klient referenci na vzdálený objekt, je tento objekt reprezentován transparentní proxy, která maskuje cílový objekt. Tato proxy dovoluje přístup k metodám cílového objektu. Pokud je volána metoda objektu reprezentovaného proxy, je tato metoda převedena na zprávu a projde příslušnými vrstvami až na server, kde je opět převedena na volání metody.



Obrázek 5.1: Architektura .NET Remotingu (převzato z (2))

5.2.1 Seřazení objektu

Zpráva přejde nejprve do serializační vrstvy (formátovač), která převede zprávu do konkrétního formátu – například SOAP. V aplikaci je použito binární formátování, které nám umožnilo dosáhnout vyšší rychlosti než při použití SOAP. Takto sestavená zpráva je předána transportnímu kanálu, který ji pomocí definovaného protokolu odešle na server. Jako transportní protokol byl použit TCP, opět z důvodu vyšší rychlosti. Na serveru přijde zpráva z transportního kanálu do formátovače, kde je

převedena na původní formát a předána dispečerovi, který zavolá požadovanou metodu objektu. Výsledek metody je stejným způsobem předán zpět klientovi. Každá z těchto vrstev (proxy, formátovač, transportní kanál) může být libovolným způsobem nahrazena za vlastní. Pro změnu jednotlivých vrstev za jinou není potřeba měnit zdrojový kód, stačí jen upravit konfigurační soubor.

5.2.2 Rozhraní

Interface neboli rozhraní je možno definovat třemi způsoby. Prvním je sdílené sestavení. V tomto případě je implementace sdíleného objektu součástí klienta. Při vytvoření instance je rozhodnuto, zda bude objekt na serveru nebo v klientovi. Výhodou tohoto způsobu je možnost lokálního použití objektu bez serveru. Při nesprávné implementaci však může dojít k problémům v souvislosti s voláním vzdálených, nebo naopak lokálních metod.

Druhým způsobem je definice pomocí sdíleného rozhraní. Na serveru se nachází kompletní implementace objektu, v klientech však pouze toto rozhraní, přes které přistupujme ke sdílenému objektu. Výhodou je jasná hranice mezi klientem a serverem.

Třetí možností je generování rozhraní z metadat sestavení. Návrh serveru je shodný s metodou sdílených sestavení. Pomocí aplikace SoapSuds lze kdykoli získat definice rozhraní ze vzdálených sestavení a s jejich pomocí vytvořit nové sestavení. Tento způsob se nedoporučuje, pouze ve velmi ojedinělých případech.

5.2.3 Typy objektů

Pro interakci mezi objekty používá remoting dva způsoby. Objekt může být předán buď referencí, nebo hodnotou.

Objekt ByValue

Objekt je nejprve seřazen (serializován) a poté celý předán. Objekty, které je potřeba předávat hodnotou, musí mít implementovat rozhraní ISerializable nebo mít atribut [Serializable]. Metody těchto objektů jsou volány lokálně.

Objekt MarshallByRef

Objekty tohoto typu jsou odvozeny od třídy MarsahllByRefObject. Objekt existuje na serveru a přijímá volání metod z klienta. Pokud klient pracuje s tímto objektem, pracuje vlastně jen s referencí, která na něj ukazuje. Tato reference není přímý odkaz

do paměti, ale ip adresa a jedinečný identifikátor právě jednoho objektu na serveru. Objekty tohoto typu se dají rozdělit do dvou kategorií: objekty aktivované serverem a objekty aktivované klientem.

Pokud klient žádá o referenci na objekt aktivovaný serverem (SAO – Server activated objects), není přenášena žádná zpráva. Až když zavolá metodu tohoto objektu, je na server poslána zpráva a ten rozhodne, zda vytvoří nový objekt (SigleCall – pro každý požadavek se vytvoří nový objekt, který po odeslání hned zanikne), anebo použije stávající (Singleton – jeden objekt na serveru obsluhuje všechny klienty).

Objekt aktivovaný klientem (CAO – Client activated object) je vytvořen na serveru v okamžiku, kdy o něj klient zažádá; poté server vrátí zpět klientovi referenci na tento objekt. Jakmile přijde tato reference na klientovu proxy, klient ji transformuje na transparentní proxy, která komunikuje s tímto objektem na serveru a dále pracuje s touto proxy.

6 SERVER

Základní úlohou serveru je udržovat model a poskytovat data všem klientům. Pokud se změní jakákoli data ve sdíleném modelu, server rozešle zprávu jako rekci na nastalou událost všem klientům, kteří mají tuto událost registrovanou. Registraci události je možno provést na jakoukoli část modelu, nicméně generování příliš mnoha událostí vyžaduje velkou režii, a proto tento postup není žádoucí. Události je potřeba registrovat pouze na části modelu, které nesou informaci o aktuálním stavu – stav světla, teplota v místnosti, atd. Popis registrace událostí je uveden v kapitole 11. Přes server jsou posílány zprávy řídicí funkce jednotlivým klientům. Tyto zprávy jsou popsány v kapitole Klient. V nynějším stavu server umožňuje spravovat jeden model a ten poskytovat většímu počtu klientů.

6.1 Konfigurace komunikace

Konfigurace komunikace je uložena v souboru „*server.exe.config*“. V tomto souboru je možné nastavit formátování zpráv (soap/binární), komunikační protokol (http/tcp), port, na kterém bude server poslouchat, adresu a typ vzdáleného objektu a další parametry remotingu. Standardně naslouchá server na portu 5555. Pro tento port je potřeba nastavit ve firewallu na serveru výjimku.

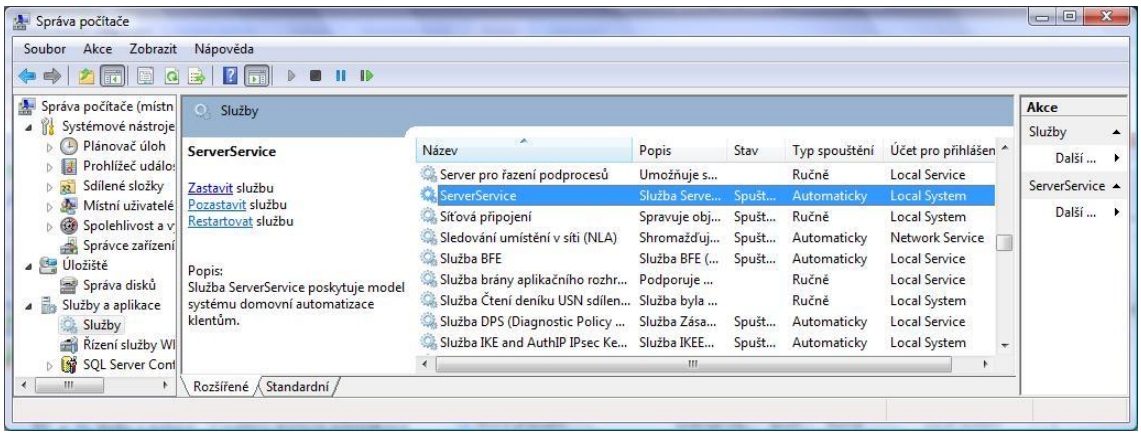
Server obsahuje funkce na správu modelů uložených v souborech. Při spuštění služby server zjistí, zda má v souboru „*temp.xml*“ uložený nějaký model. Pokud tomu tak je, nahraje ho. V případě, že soubor neexistuje nebo ho není možné načíst, nabídne klientům seznam všech modelů uložených na serveru. Z tohoto seznamu může klient vybrat jakýkoli soubor pro nahrání aktuálního modelu.

Server je navržen jako služba systému Windows. Služba systému je program, který běží dlouhou dobu na pozadí, nemá žádné grafické rozhraní a nepotřebuje zásah uživatele. Služby mohou být nakonfigurovány tak, aby se automaticky spouštěly po startu systému. Službu není možné spustit přímo, je potřeba ji nejprve nainstalovat.

Instalace se provádí pomocí nástroje InstallUtil standardně umístěného v C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe .

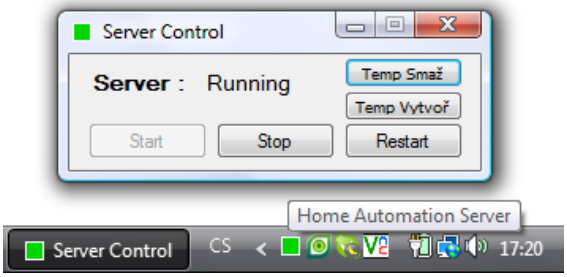
```
InstallUtil /i serverservice.exe
```

Odinstalování služby se provede stejným příkazem, pouze se změní parametr na "*/u*". Veškeré nainstalované služby lze procházet, spouštět a zastavovat pomocí Správy počítače. Obrázek 6.1 popisuje okno Správy počítače s vyznačenou službou serveru.



Obrázek 6.1: Nainstalované služby systému Windows

Protože služba nemůže mít žádné grafické rozhraní, je velmi užitečné mít nástroj pro ovládání. Pro tyto účely byl vytvořen nástroj Server Control. Obrázek 6.2 ukazuje grafické rozhraní kontrolního nástroje. Okno obsahuje popis aktuálního stavu serveru a tři tlačítka pro rychlé ovládání ve spodní řadě. Tlačítka „*Temp smaž*" a „*Temp vytvoř*" slouží pro vytvoření nebo smazání souboru „*temp.xml*", který se načítá při startu serveru. Nástroj může být zobrazen pouze v Task Baru (první ikona zleva), kde barva ikony určuje aktuální stav serveru.



Obrázek 6.2: Kontrolní nástroj pro serverovou službu

7 KLIENT

Klientem se rozumí jakákoli aplikace přistupující k serveru přes .NET remoting. Aplikace komunikující přímo s hardwarem je klient, stejně jako grafické rozhraní použité pro konfiguraci.

7.1 Konfigurace komunikace

Všichni klienti, bez ohledu na funkci, se připojují k severu přes stejné rozhraní. Konfigurace připojení je obdobně jako u serveru umístěna v souboru „*client.exe.config*“. Zde je potřeba nastavit stejné formátování a shodný přenosový protokol jako na serveru a zadat adresu objektu. Tato adresa musí být v „*URL*“ formátu, to znamená, že musí obsahovat protokol, ip adresu, port a jméno objektu. Při odesílání událostí posílá server klientovi zprávy na určitý port, který je opět nastaven v tomto souboru. Pro tento port musí být nastavena výjimka ve firewallu klienta. Pokud bude klient za NATem bez nastaveného port forwardingu, nebo bude firewall bez udělené výjimky, nebudou klientovi přicházet asynchronní události od serveru. Klient se při příjmu asynchronních událostí chová jako server, který naslouchá na definovaném portu. Na tento port přicházejí zprávy o změně modelu. Z tohoto vyplývá, že systém bude možné konfigurovat, vizualizace však nebude dostupná online. Tento problém bude potřeba obejít úpravou transportního kanálu remotingu.

7.2 Zprávy

V systému existují dva druhy zpráv. První jsou oznámení událostí. Jsou generovány automaticky při změně jakékoli proměnné v modelu, na kterou je nějaká událost registrovaná. Oznámení se zašle všem klientům, kteří mají tuto událost registrovanou, a oni zareagují zavoláním příslušné obsluhy události.

Tabulka 7.1: Zprávy pro řízení klientů

Zpráva	Akce klienta připojeného k HW
DownloadModel	Stáhne model z HW a nahraje na server
UploadModel	Stáhne model ze serveru a nahraje do hardware
UploadDevice	Stáhne jednu komponentu ze serveru a nahraje do hardware. Reference na komponentu je parametr zprávy.

Zprávy druhého typu jsou řídicí zprávy. Slouží ke vzájemnému řízení serveru a klientů. Tyto zprávy jsou generovány pouze při požadavku uživatele. V aktuální verzi jsou používány výše zmíněné tři zprávy, ale v příští verzi softwaru se jejich počet rozšíří. Potřeba jsou zprávy nařizující serveru určité akce (například restart) a informační zprávy, které budou poskytovat informace o chystaném restartu serveru, o nahrazení celého modelu jiným klientem, o stavu připojeného driveru, o stavu hardwaru, atd. Tabulka 7.1 obsahuje seznam všech zpráv, které jsou generovány v grafickém uživatelském prostředí (GUI - Graphic User Interface) po akci uživatele. Z GUI je zpráva zaslána na server, ten ji předá všem klientům a driver, tedy klient, který je připojený k příslušnému hardware, vykoná příslušnou akci.

8 SYSTÉM DAMIC

Systém Damic je distribuovaný systém domovní automatizace, vyvinutý na katedře řídicí techniky Fakulty elektrotechniky ČVUT v Praze. Systém je navržen tak, aby uživateli poskytoval maximální komfort a jednoduché nastavení. Technicky zdatný uživatel by měl být schopen systém sám zapojit a nakonfigurovat na svém PC.

Kapitola byla napsána za pomocí zdrojů (5),(6),(7),(8),(9),(10),(11); podkapitola uLan napsána pomocí zdroje (12).

8.1 uLan

ULan je devítibitový komunikační protokol využívající fyzickou vrstvu linky RS 485. Znaky jsou přenášeny stejným způsobem jako na RS 232 při asynchronním přenosu až na paritní bit, který je použit k rozlišení datových znaků a řídících znaků. Použití devítibitových znaků zjednodušuje přenos binárních dat a inteligentním kontrolérům může snížit zatížení procesoru, protože procesor nemusí zpracovávat data určená jinému zařízení. Řada výrobců mikrokontrolérů nabízí ve většině svých procesorů devítibitové rozšíření UARTu.

8.1.1 Formát datového rámce

Datový rámec je základní jednotka protokolu uLan. V rámci je uveden zdroj, cíl, typ rámce nebo příkaz, data, ukončující znak a kontrolní součet. Rámec se skládá ze sekvence devítibitových znaků. Znaky jsou přenášeny asynchronně; z toho vyplývá, že každý znak má start bit na začátku a stop bit na konci. Řídicí znaky mají devátý bit nastaven na jedna a mohou se vyskytovat pouze na začátku a na konci rámce.

Obrázek 8.1: Formát datového rámce uLanu (podle (15))

Obrázek 8.1 reprezentuje popis formátu datového rámce. Datový rámec začíná buď řídicím znakem, který určuje cíl rámce, nebo znakem určujícím odpověď. Tento znak přijmou všechna zařízení a rozhodnou, zda budou přijímat daný rámec. Za druhým znakem, který určuje cílovou adresu, následuje číslo příkazu nebo typ zprávy. Poté

jsou v rámci umístěny datové bajty a za nimi ukončující znak, který určuje, jak se data mají v přijímacím zařízení zpracovat. Posledním znakem je kontrolní součet.

8.1.2 Přístup k médiu

ULan používá pro přístup k médiu deterministické distribuované přiřazení média, kterého je dosaženo použitím časových pravidel pro přístupovou sekvenci. Sekvence se skládá z přerušovacích znaků (11 nulových bitů), čekání po určitý interval a naslouchání přerušovacích znaků přicházejících od ostatních zařízení. Všechna zařízení musejí čekat po dobu čtyř až dvaceti znaků. Tato doba je určena z rozdílu vlastní adresy a adresy posledního zařízení, které mělo přiřazené médium. Toto schéma snižuje pravděpodobnost kolize a zároveň vede k cyklické změně priority jednotlivých zařízení.

Tabulka 8.1: Formát zprávy pro práci s objekty

		Číslo byte	Význam	
			Hlavička	
		0	BCMD	ID příkazu, který bude mít odpověď
		1	BSN	sériové číslo, které bude mít odpověď
		2	SN	sériové číslo požadavku
0 – MAXD			**Příkaz**	
		0	Dolní byte příkazu	příkaz, který se má po doručení zprávy vykonat; bez příkazu = zápis
		1	Horní byte příkazu	viz. Tabulka 8.2;
			Tělo	Tělo zprávy se může opakovat až MAX-krát
	0 - MAX	0	Dolní byte adresy OID	adresa OIDu
		1	Horní byte adresy OID	
		2	data 0	data OIDu, délka je podle podle typu dat pro daný OID
			...	
		n - 2	data n	
			Teminátor	
		0	0x00 nebo 0x14	pokud je vyžadována odpověď, použije se v nultém bytu 0x14, jinak 0x00
		1	0x00	

8.1.3 Objektový slovník

Každé zařízení na uLanu obsahuje takzvaný objektový slovník. Tento slovník obsahuje veškeré informace nutné pro nastavení a řízení zařízení. Slovník je pole všech proměnných, které jsou přístupné přes uLan. Slovník obsahuje číslo, název, typ a atributy proměnné. Jedna proměnná ze slovníku se nazývá OID.

Tabulka 8.2: Seznam příkazů pro práci s objektovým slovníkem (převzato z (13))

Název	Číslo	Číslo (HEX)	Parametr	Popis
I_AOID	10	0A	string	Nahradí číselný OID textovým názvem
I_DOII	12	0C	oid	Dotaz na popis OIDu pro zápis(vstupního)
I_DOIIr	13	0D	oidoiddesc	Popis OIDu pro zápis(vstupního)
I_DOIO	14	0E	oid	Dotaz na popis OIDu pro čtení(výstupního)
I_DOIOr	15	0F	oidoiddesc	Popis OIDu pro čtení(výstupního)
I_QOII	16	10	oidmaxret	Dotaz na seznam OIDů pro zápis(vstupních)
I_QOIIr	17	11	OID(0)... OID(n)	Seznam OIDů pro zápis(vstupních)
I_QOIO	18	12	oidmaxret	Dotaz na seznam OIDů pro čtení(výstupních)
I_QOIOr	19	13	OID(0)... OID(n)	Seznam OIDů pro čtení(výstupních)
I_RDRQ	20	14	OID(0)... OID(n)	Dotaz na hodnoty OIDů uvedených v parametru
I_RDRQr	21	15	OID(0)... OID(n)	Odpověď na dotaz na hodnoty
I_SNCHK	29	1D	SN0 .. SN3	Dotaz na sériové číslo zařízení
I_STATUS	30	1E		Dotaz na 16 bitový stavový registr
I_ERRCLR	31	1F		Resetuje STATUS ERROR

8.1.4 Zprávy pro práci s objekty

Protokol uLan zahrnuje příkazy pro práci s objektovým slovníkem a jednotlivými OIDy. Tabulka 8.1 obsahuje popis formátu zprávy pro operace s objektovým slovníkem a jednotlivými OIDy. Každý OID reprezentuje konkrétní parametr zařízení, například stav výstupu, stav vstupu, atd. Každá zpráva začíná hlavičkou, ve které je uveden identifikátor zprávy, sériové číslo odpovědi a sériové číslo zprávy. Následuje příkaz (Tabulka 8.2 uvádí seznam jednotlivých příkazů) a v těle jsou uvedeny parametry příkazu. Pokud není příkaz uveden, uLan interpretuje následující data jako parametry zápisu. Parametry mohou být například OIDy, které chceme zapsat nebo přečíst. Počet parametrů je omezen pouze maximální délkou zprávy. Tato délka se může lišit v závislosti na konkrétním zařízení, a proto je doporučeno používat maximální délku zprávy do 1kB. Délku odpovědi je nutno odhadnout předem a podle toho upravit dotaz. Zpráva je ukončena terminátorem, kde je uvedeno, zda požadujeme odpověď na zprávu či nikoli. Trojici dat „příkaz, tělo, terminátor“ lze za

sebe vhodným způsobem řetězit při dodržení maximální délky zprávy. V jedné zprávě tak lze provést naráz čtení a zápis.

8.2 Prvky systému

Všechny prvky jsou rovnocenné, propojené pomocí sériové linky RS 485, a komunikují pomocí protokolu uLan. Pokud chceme například rozsvítit světlo, zmáčkneme tlačítko na vypínači a ten odešle přiřazenou zprávu po sběrnici. V této zprávě je akce, která se má provést, a cílové zařízení. Toto zařízení přijme zprávu a vykoná akci uvedenou ve zprávě – v našem případě rozsvícení světla.

8.2.1 Pokojový regulátor uLTH

Jednotka pokojového regulátoru je určena k optimálnímu řízení teploty v místnosti. Regulátor může být vybaven displejem pro zobrazování teploty a aktuálního topného režimu, popřípadě tlačítky pro nastavení těchto parametrů. Souhrn - standardní vybavení, senzor teploty a dva digitální vstupy/výstupy - je možno rozšířit o externí čidlo teploty nebo senzor vlhkosti. Na digitální vstupy/výstupy lze připojit spínání termohlavic, okenní kontakty nebo spínání ventilace. Externí senzor teploty může sloužit k měření teploty podlahového vytápění, venkovní teploty nebo teploty na jiném místě pokoje, či v jiném pokoji.

Regulátor umožňuje nastavit několik teplotních schémat a podle nich regulovat teplotu v průběhu dne. Teplotní křivky, včetně schémat, jsou plně nastavitelné z grafického rozhraní na PC, stejně jako veškeré ostatní nastavení. Je také možno nastavit je přímo na termostatu. Pokud je na vstup připojen okenní kontakt, regulátor automaticky vypne topení, když je otevřené okno.

8.2.2 Pokojový vypínač uLSW

Vypínač je realizován ve dvou nebo čtyř tlačítkovém provedení. Každé tlačítko má čtyři funkce: stisk, dvojstisk, dlouhý stisk nebo stisk s nastavitelnou délkou. Každé této funkci lze přiřadit libovolnou akci systému. Například na jeden spínač je možné nastavit libovolnou akci na kterýkoli akční člen v systému. Konfigurace se opět provádí v grafickém rozhraní na PC. Popis této konfigurace je v kapitole Grafické uživatelské rozhraní.

8.2.3 Akční člen uACT

Akční člen uACT je vybaven dvěma reléovými výstupy pro spínání až 230 VAC. K těmto vstupům může být připojeno spínání termohlavic, světel, žaluzií nebo ventilátorů. Dále obsahuje dva digitální vstupy pro připojení okenního kontaktu nebo koncových spínačů žaluzií a vstup pro připojení čidla teploty nebo vlhkosti. Akční člen může obsahovat samostatný program, který zastoupí pokojový regulátor v případě poruchy.

8.2.4 Akční člen uLMO

Akční člen uLMO je levnější variantou uACTu. Obsahuje stejné vstupy, pouze reléové výstupy jsou určené pouze na 24 VDC. Chování je také shodné s uACTem.

8.2.5 Stmívač uDIM

Stmívač uDIM je určen pro ovládání až osmi světel. Stmívač podporuje všechny požadované funkce na komfortní ovládání světel: okamžité rozsvícení/zhasnutí, postupné zvýšení nebo snížení jasu na plnou nebo zadanou intenzitu, nastavení aktuální intenzity světla, atd. Ovládání těchto funkcí lze přiřadit libovolnému vypínači opět přes GUI na PC. V kombinaci s pokojovým regulátorem, může jakékoli světlo plnit funkci světelného budíku – v danou dobu se bude postupně rozsvěcet na danou intenzitu.

8.2.6 Digitální vstupy uLMI

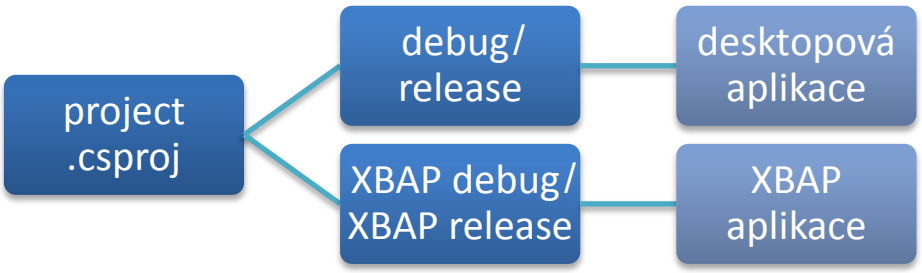
Modulu digitálních vstupů lze využít jako doplňku k ostatním zařízením v případě, že daný počet vstupů nestačí. Obsahuje 4 nebo 6 digitálních vstupů a volitelný vstup pro čidlo teploty, která může být použita jako doplňková informace pro pokojový regulátor. Vstupy mohou být použity pro snímání koncových spínačů žaluzií, signalizace otevřených dveří či oken a podobně.

9 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

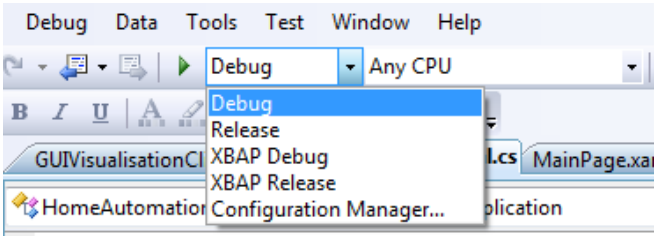
Grafické rozhraní slouží pro komfortní konfiguraci, správu, nastavení nebo ladění systému Damic. Byly navrženy dvě samostatné aplikace – klient pro koncového uživatele s plnou vizualizací systému a možností konfigurace a klient pro přímou editaci modelu, který umožňuje model jednoduše procházet a zapisovat do něj přímo na konkrétní pozice.

9.1 Klient pro koncového uživatele

Klient pro koncového uživatele s grafickým rozhraním byl navržen s knihovnami pro systém Damic. Umožňuje tento systém plně konfigurovat a vizualizovat. Aplikace je dostupná ve dvou verzích. První verze je aplikace desktopová a druhá je tzv. XBAP verze, spustitelná přes webový prohlížeč. Aby bylo možné takto aplikaci kompilovat, je potřeba do Visual Studia přidat příslušný template. Velmi dobrým se ukázal template „*Flexible Application*“, popsaný v (14). Jednoduchým přepnutím konfigurace se celý projekt sestaví jako jiný typ aplikace, viz Obrázek 9.2. Obrázek 9.1 popisuje postup, jak přepnout konfiguraci.



Obrázek 9.2: Vytvoření desktopové a XBAP aplikace ze stejného projektu, převzato z (14)



Obrázek 9.1: Přepnutí konfigurace na XBAP

9.1.1 Windows Presentation Foundation

Kapitola byla napsána za použití zdroje (15),(2).

Pro grafické rozhraní byla použita technologie Windows Presentation Foundation (WPF). Objevila se v .NET Frameworku 3.0 a je standardní součástí systémů Windows Vista a Windows Server 2008. Má podporu i ve starších Windows XP SP2 a Windows Server 2003 pro instalaci příslušných aktualizací. V grafickém klientovi pro systém Damic jsou použity součásti .NET Frameworku 3.5 SP1, který je nutný pro korektní spuštění aplikace.

WPF používá pro popis grafických prvků značkovací jazyk (markup language) XAML (Extensible Application Markup Language). Tímto jazykem je možné popsat a navrhnout bohaté grafické komponenty se spoustou ozdobných prvků. Pro návrh jednotlivých komponent byl použit Microsoft Expression Blend 2.0. Bohužel, tento nástroj neobsahuje žádnou podporu pro psaní zdrojového doplňkového kódu. V Microsoft Visual Studiu 2008 zase naopak chybí větší podpora XAML.

WPF využívá pro vykreslování 2D a 3D grafiky přímo DirectX. Všechny současné běžně používané grafické karty mají hardwarovou podporu DirectX 7 a výše. WPF tedy přímo využívá grafické karty pro vykreslování jednotlivých grafických prvků, a tím dosahuje vysoké rychlosti zpracování.

Velkou výhodou aplikací napsaných v XAML je, že mohou být zkompilovány tak, aby byly spustitelné buď jako desktopová aplikace, anebo jako XBAP (XAML Browser Application) aplikace, která se spouští prostřednictvím internetového prohlížeče. XBAP nepovoluje z bezpečnostních důvodů některé funkce systému. Aby bylo možné využívat v XBAP stejné funce jako v desktopové verzi, musí běžet v tzv. „Full Trust“ modu. Aplikace musí být v tomto modu podepsána elektronickým podpisem a uživatel, který chce aplikaci spustit, musí mít nainstalovaný příslušný certifikát. Rámec XBAP aplikací a bezpečnosti překračuje rozsah této práce, nicméně více lze najít v Pro WPF (2), v kapitole XAML Browser Application. XBAP podporují prohlížeče Internet Explorer,verze 7 a vyšší, a Mozilla Firefox, verze 3 a vyšší.

9.1.2 Přehled funkcí

Klient má význam jak pro technika, tak i pro uživatele. Umožňuje uložit model do souboru, načíst model ze souboru, nahrát model na server a uložit model do souboru na server. Pokud je z hardwaru stažena konfigurace, je možné ji změnit a nahrát zpět.

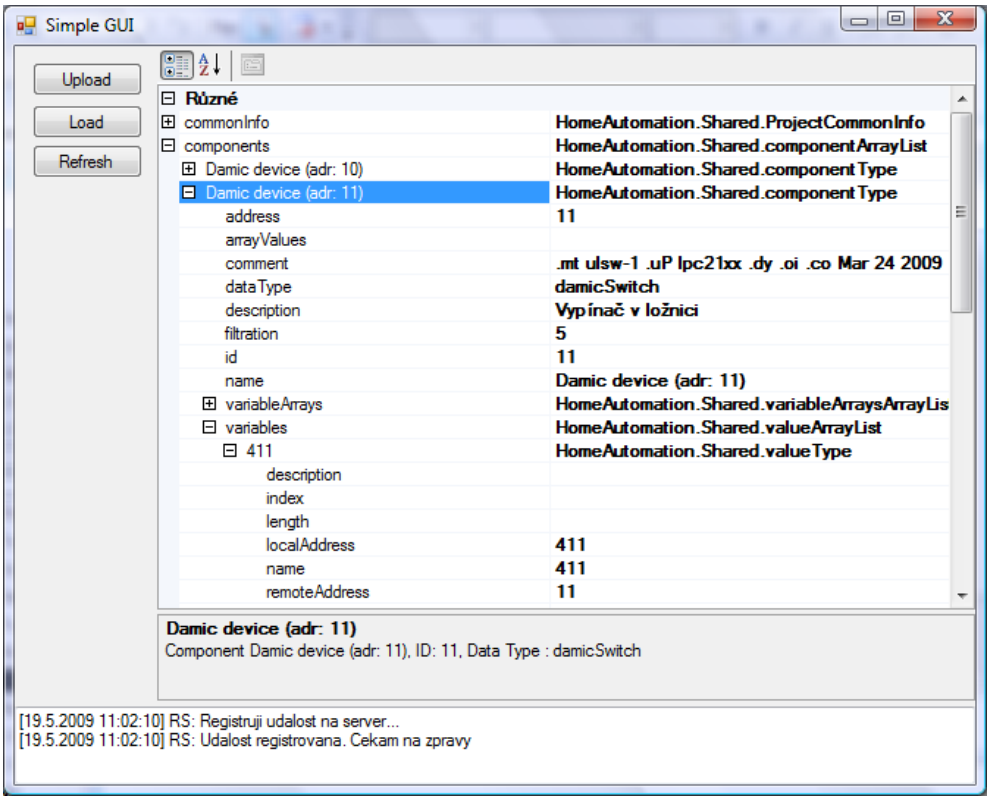
Vizualizace zobrazuje jednotlivé stavy systému s definovanou časovou prodlevou. Obrázek 9.3 schematicky popisuje jednotlivé funkce systému. Horní obrázek – úvodní obrazovka - obsahuje navigaci do jednotlivých částí klienta. Zleva je to Konfigurace systému, Zobrazení teplot, Nastavení časových plánů a Vizualizace. Popis, jak systém nastavit a jak jednotlivé části fungují, lze najít v příloze „Stručný úvod do konfigurace systému Damic“.

Obrázek 9.3: Schematický popis grafického rozhraní pro koncového uživatele

9.2 Klient pro přímou editaci modelu

Především pro ladící účely byl vytvořen klient pro přímou editaci a procházení modelu. Lze ho ale použít i k přímému nastavení jednotlivých parametrů systému. Obrázek 9.4 takového klienta popisuje. Okno obsahuje stromovou strukturu modelu umístěného na serveru, přesně podle definice modelu. Při procházení struktury se všechna data načítají přímo ze serveru. Pokud je některý údaj změněn, okamžitě se změna projeví i v modelu na serveru. Pokud je změněna některá z proměnných, na

kterou má jiný klient zaregistrovanou událost, pak tato událost nastane. Tlačítka vlevo slouží k nahrání modelu ze serveru do hardwaru, stáhnutí konfigurace z hardwaru do modelu a k obnovení adresářové struktury, pokud dojde k výpadku spojení.



Obrázek 9.4: Klient pro přímou editaci modelu

10 DRIVER PRO DAMIC

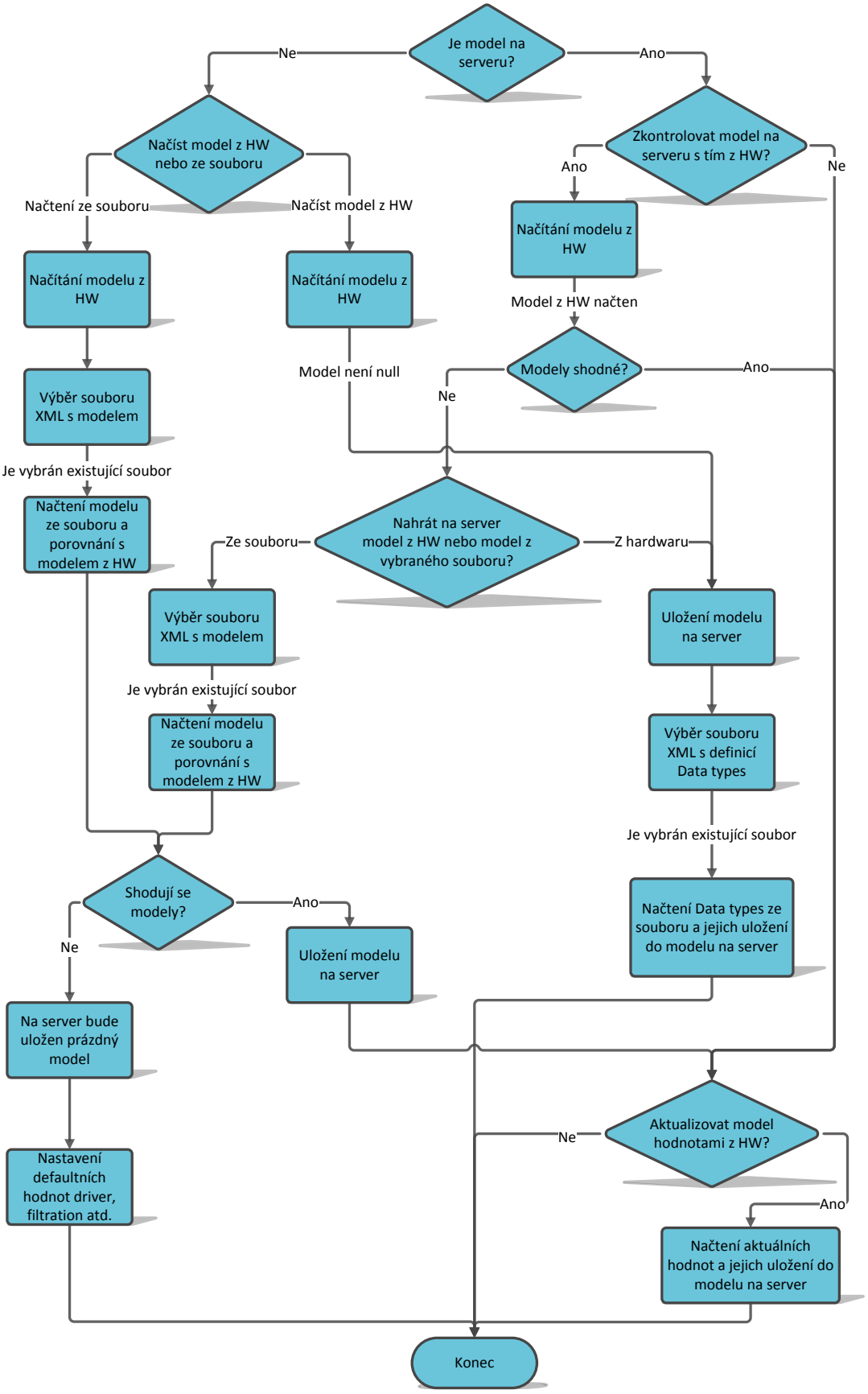
Klient, který přistupuje přímo k hardwaru, se nazývá driver. Tento klient umí zjistit všechna dostupná zařízení, načíst jejich konfiguraci a z těchto informací vytvořit kompletní model. Model lze nahrát na server nebo uložit do souboru. Pokud je model na serveru, driver k němu přistupuje vzdáleně a zapisuje do něj změny stavových proměnných. ULan v dnešní podobě neumožňuje informovat ostatní zařízení o změně stavu v některém zařízení, proto je nutné všechny stavové veličiny číst cyklicky ve stanoveném časovém intervalu. Tento interval by neměl být delší než pět vteřin, protože delší prodleva je pro uživatele ve vizualizaci nekomfortní.

Driver může být spuštěn v automatickém nebo manuálním režimu. Automatický režim (Obrázek 10.1 vlevo) zjistí nejprve na serveru, zda je model dostupný. Obrázek 10.2 popisuje algoritmus načítání modelu na server.

V manuálním režimu (Obrázek 10.1 uprostřed) jsou jednotlivé kroky přiřazené příslušným tlačítkům a uživatel má plnou kontrolu nad aktuálním průběhem načítání. Tento režim se hodí pouze pro účely ladění, pro běžného uživatele nebude dostupný.

Poslední záložka (Obrázek 10.1 vpravo) je logovací konzola, na kterou se zapisují stavy a průběhy jednotlivých akcí.

Obrázek 10.1: Okno klienta s driverem pro Damic (postupně každá záložka)

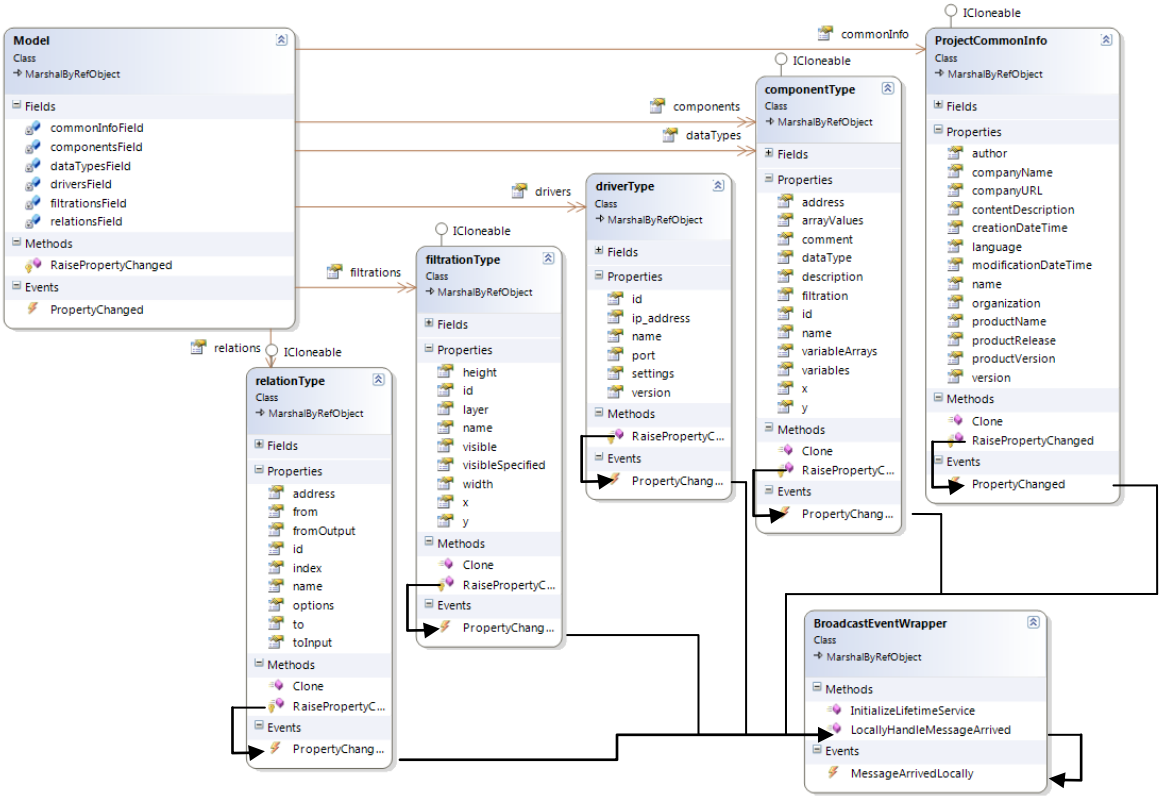


Obrázek 10.2: Popis postupu načítání modelu z hardwaru na server

11 POPIS JEDNOTLIVÝCH ČÁSTÍ KÓDU PRO DALŠÍ VÝVOJ APLIKACE

11.1 Model

Model je implementován podle specifikace popsané v kapitole 4. Jednotlivé třídy odpovídají definicím v XSD souboru. Obrázek 11.1 zobrazuje popis základního modelu. Změna jakékoli vlastnosti v kterékoli části modelu způsobí aktivaci metody „*RaisePropertyChanged*“, která vyvolá událost „*PropertyChanged*“ a ta zavolá metodu „*LocallyHandleMessageArrived*“ ze třídy „*BroadcastEventWrapper*“. Tím nastane událost „*MessageArrivedLocally*“, na niž jsou registrovány obsluhy událostí změn jednotlivých klientů. Tento způsob předávání událostí je nutný kvůli korektnímu přenosu přes .NET Remoting. Bez mezikroku přes třídu „*BroadcastEventWrapper*“ by se server pokoušel zavolat obsluhu události ze sestavení klienta, která ovšem na serveru není dostupná, a tak by došlo k chybě. Nejnižší zpoždění při předání zprávy od jednoho klienta přes server ostatním klientům je okolo 20ms na lokální ethernetové síti.



Obrázek 11.1: Reprezentace modelu jednotlivými třídami

Delegát události „*MessageArrivedLocally*“ je „*ModelPropertyChangedEventHandler*“. Registraci události v klientovi popisuje následující ukázka. Nejprve je vytvořena instance „*eventWrapper*“ třídy „*BroadcastEventWrapper*“, které se na událost „*MessageArrivedLocally*“ zaregistruje lokální funkce klienta „*HandleMessage*“. Vydálenému modelu „*Model*“ se na událost „*MessageArrived*“ zaregistruje metoda „*eventWrapper.LocallyHandleMessageArrived*“. Tímto postupem se budou události zcela korektně předávat

```
BroadcastEventWrapper eventWrapper = new BroadcastEventWrapper();
eventWrapper.MessageArrivedLocally += new
             ModelPropertyChangedEventHandler(HandleMessage);
Model.MessageArrived +=
new ModelPropertyChangedEventHandler(eventWrapper.LocallyHandleMessageArrived);
```

V modelu je pro každé pole hodnot použita třída zděděná od třídy „ArrayList“. Každá z těchto příd má implementované rozhraní „*ICustomTypeDescriptor*“. Třídy s tímto rozhraním lze procházet jako stromovou strukturu v komponentě „*PropertyGrid*“. Tyto ArrayListy jsou vytvořeny pro každou třídu jednotlivě, právě z důvodu korektního procházení a možnosti každou třídu popsat zvlášť. Konkrétně se jedná o ArrayListy pro komponenty, hodnoty, divery, filtrační kategorie, relace a dvourozměrná pole hodnot.

11.1.1 Sestavení General

Sestavení „*General*“ obsahuje všechny třídy potřebné pro práci s modelem lokálně. Tato knihovna je součástí všech aplikací, které pracují s modelem. Obsahuje všechny součásti popsané v minulé kapitole, a navíc rozhraní „*ISharedModel*“ pro přístup k modelu vzdáleně. Přes toto rozhraní přistupují klienti k metodám a datům na serveru. V rozhraní jsou obsaženy funkce pro práci:

- s celým modelem
  - načítání a ukládání do souboru
  - načítání a ukládání do „streamu“
- se seznamem modelů
- s jednotlivými částmi modelu
  - drivery
  - komponenty
  - datové typy
  - hodnoty
  - atd.
- se zprávami pro řízení ostatních klientů nebo serveru

11.1.2 Sestavení SharedModel

Třída „*SharedModel*“ implementuje rozhraní „*ISharedModel*“ a je potomkem třídy „*Model*“. Knihovna vzniklá ze sestavení „*SharedModel*“ je součástí pouze serveru a vzdáleně se k ní přistupuje přes rozhraní „*ISharedModel*“.

Popis rozhraní ISharedModel

- Metody pro práci se streamem
  - LoadModelFromStream – načte model ze streamu
  - GetStream – návratová hodnota je stream s celým modelem
- Metody pro práci se soubory
  - GetModelFilesList – vrátí seznam všech souborů uložených na serveru
  - SaveModelToFile – uloží model do souboru na serveru
  - LoadModelFromFile – načte model ze souboru na serveru
- Metody pro posílání zpráv
  - BroadcastMessage – rozešle zprávu všem klientům. Zpráva může obsahovat odesílatele, typ zprávy a parametry. Touto metodou jsou zasílány řídicí zprávy.
- Práce s celým modelem
  - GetModel - vrátí ukazatel na Model
- Práce s jednotlivými částmi modelu – pro každou součást modelu jsou implementovány metody „*Get*“ a „*Set*“ a inicializační metody. Pro třídy „*ComponentType*“ a „*ValueType*“ metody hledání podle definovaných parametrů.

Metoda „*BroadcastMessage*“ se pokusí odeslat zprávu všem klientům, kteří mají zaregistrovanou událost pro příjem řídicích zpráv „*OnHandleMessage*“. V případě, že se některý z klientů odpojí a neodebere registrovanou událost, dojde na tomto volání k chybě. V reakci na tuto chybu je odebrán příslušný klient ze seznamu dostupných klientů.

11.2 Server

Server je napsán ve dvou verzích –„ aplikace příkazového řádku“ a „služba systému Windows“, která je pojmenována „*ServerService*“. Verze pro příkazový řádek je určena pouze pro ladící účely.

„*ServerService*“ se skládá ze dvou tříd. Třída „*ServerServiceInstaller*“ je odvozena od třídy „*Installer*“. V této třídě se definují jednak vlastnosti služby, jednak způsob, jak se má služba instalovat do systému. Pokud by tato třída chyběla, nebylo by možné službu korektně nainstalovat.

Třída obsahující funkcionalitu serveru „*ServerService*“ je odvozena od třídy „*ServiceBase*“. Tato třída obsahuje obsluhu základních systémových zpráv pro služby jako je „*Start*“, „*Stop*“, „*Pause*“, „*Continue*“, „*Shutdown*“ a uživatelské zprávy, které jsou definovány pouze číslem. Číslo uživatelských zpráv může být v rozsahu 128 až 255. Uživatelské zprávy jsou definovány dvě – uložení aktuálního modelu do souboru „*temp.xml*“ (číslo 160) a smazání souboru „*temp.xml*“ (číslo 150). Pokud tento soubor existuje při startu serveru, je automaticky načten a poskytnut klientům.

Při startu serveru je načtena konfigurace .NET remotingu z konfiguračního souboru „*server.exe.config*“. Struktura je popsána v následující ukázce.

```
<configuration>
  <system.runtime.remoting>
    <application>
      <channels>
        <channel ref="tcp" port="5555">
          <serverProviders>
            <formatter ref="binary" typeFilterLevel="Full" />
          </serverProviders>
          <clientProviders>
            <formatter ref="binary" />
          </clientProviders>
        </channel>
      </channels>
      <service>
        <wellknown mode="Singleton" type="HomeAutomation.Shared.SharedModel,
                                  SharedModel" objectUri="SharedModel.soap" />
      </service>
      <lifetime leaseTime="0" renewOnCallTime="2M" />
    </application>
  </system.runtime.remoting>
</configuration>
```

Definice přenosového protokolu a portu, na kterém server poslouchá

Binární formátování na serveru i klientovi. typeFilterLevel musí být Full, aby se korektně formátovaly i události.

Mód objektu je Singelton, typ poskytovaného objektu je popsán jako „typ, sestavení“ a identifikátor objektu objectUri definuje, kam se budou připojovat klienti

Definuje dobu existence objektu – po celou dobu běhu serveru bude objekt k dispozici.

11.3 Klient

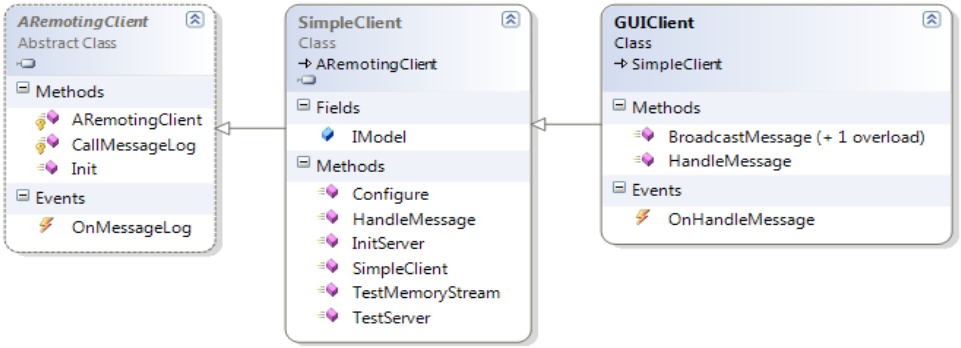
Klient je jakákoli aplikace, která se připojuje k serveru přes .NET Remoting. Pro snadnou implementaci dalších klientů je vytvořena hierarchie tříd, popsaná v následujících kapitolách.

11.3.1 Popis tříd

Základní třídy „*ARemotingClient*“ a „*SimpleClient*“ jsou v sestavení „*AClient*“. Toto sestavení je ve formě dynamicky připojované knihovny (DLL) součástí všech klientů. Třída „*ARemotigClient*“ obsahuje veřejnou metodu „*Init*“, která načte konfiguraci .NET remotingu ze souboru daného parametrem této metody. Metoda „*CallMessageLog*“ volá událost „*OnMessageLog*“, na kterou je registrováno zpracování záznamů v odvozených třídách.

Třída „*SimpleClient*“ rozšiřuje třídu „*ARemotingClient*“ o práci s událostmi a modelem. Metoda „*Configure*“ inicializuje vlastnost „*IModel*“ typu „*ISharedModel*“ tak, aby ukazovala na vzdálený model na serveru, a zároveň zaregistruje metodu „*HandleMessage*“ pro příjem řídicích zpráv ze serveru. Inicializace modelu na serveru na prázdný model se provádí metodou „*InitServer*“. Ostatní metody začínající slovem „*Test*“ jsou pouze pro ladící účely nebo pro testování rychlosti. Obrázek 11.2 popisuje hierarchii základních tříd klienta.

Základní třídou pro grafické klienty je „*GUIClient*“. Rozšiřuje třídu „*SimpleClient*“ o událost „*OnHandleMessage*“, která nastane při příjmu řídicí zprávy a o metodu „*BroadcastMessage*“, která implementuje odesílání řídicích zpráv.



Obrázek 11.2: Hierarchie tříd klienta

11.3.2 Konfigurační soubor

Konfigurační soubor pro klienta je velmi podobný souboru pro server. Rozdíl je hlavně v definici objektu. K objektu se přistupuje přes rozhraní „*ISharedModel*“ v sestavení „*General*“, znamená to tedy, že sestavení „*SharedModel*“, které je použito na serveru, není již potřeba. URL adresa se skládá z protokolu, který je použit pro komunikaci se serverem (stejný jako v atributu „*channel*“), ip adresy serveru, portu, na kterém server naslouchá (nastavený v konfiguračním souboru serveru), a vzdáleného objektu (opět uvedený v konfiguračním souboru serveru). Význam ostatních parametrů je stejný jako v konfiguračním souboru serveru.

```
<configuration>
  <system.runtime.remoting>
    <application>
      <channels>
        <channel ref="tcp" port="5556" >
          <clientProviders>
            <formatter ref="binary" typeFilterLevel="Full" />
          </clientProviders>
          <serverProviders>
            <formatter ref="binary" typeFilterLevel="Full" />
          </serverProviders>
        </channel>
      </channels>
      <client>
        <wellknown type="HomeAutomation.Shared.ISharedModel, General"
       url="tcp://147.32.86.102:5555/SharedModel.soap" />
      </client>
      <lifetime leaseTime="0" renewOnCallTime="2M" />
    </application>
  </system.runtime.remoting>
</configuration>
```

Klient poslouchá příchozí zprávy od serveru na portu 556 a komunikace probíhá přes protokol TCP

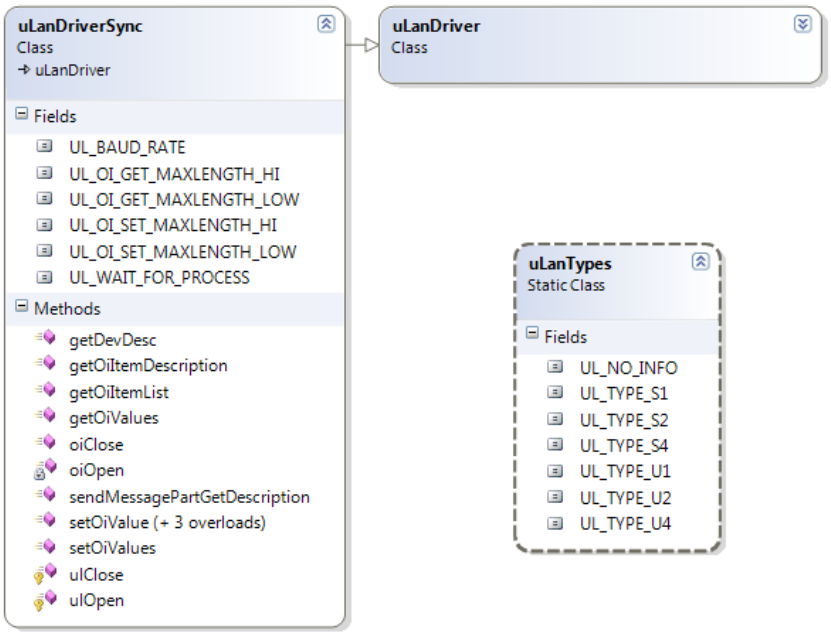
Klient přistupuje k modelu na serveru přes rozhraní ISharedModel v sestavení General. Objekt modelu je umístěn na definované URL.

11.4 Driver

Driver je jakýkoli klient, který má přímý přístup k hardwaru, umí z něj vytvořit model a ten zpětně nahrát do hardware. Drivery byly napsány pro systém Damic (uLan) a systém domovní automatizace s PLC Wago (Modbus).

11.4.1 uLan

Pro přístup k uLanu je použit převodník USB – uLan. Knihovna, která zprostředkovává přístup k funkcím uLanu, je „*libulan.dll*“. Tato knihovna je importována do projektu prostřednictvím třídy „*uLanDriver*“. Třída „*uLanDriverSync*“ rozšiřuje třídu „*uLanDriver*“, využívá její funkce a zjednodušuje práci s uLanem. Všechny vlastnosti třídy „*uLanDriverSync*“ nastavují parametry uLan komunikace. Metody třídy slouží k otevření a ukončení komunikace, získání popisu zařízení, popisu jednotlivých OIDu, seznamu OIDu, hodnot OIDu a zápisu příslušných hodnot. Systém Damic používá prozatím pouze šest typů hodnot: jednobajtový, dvoubajtový a čtyřbajtový integer se znaménkem a bez znaménka. Seznam těchto typů je popsán ve třídě „*uLanTypes*“.



Obrázek 11.3: Schéma tříd pro přístup k uLanu

11.4.2 WAGO

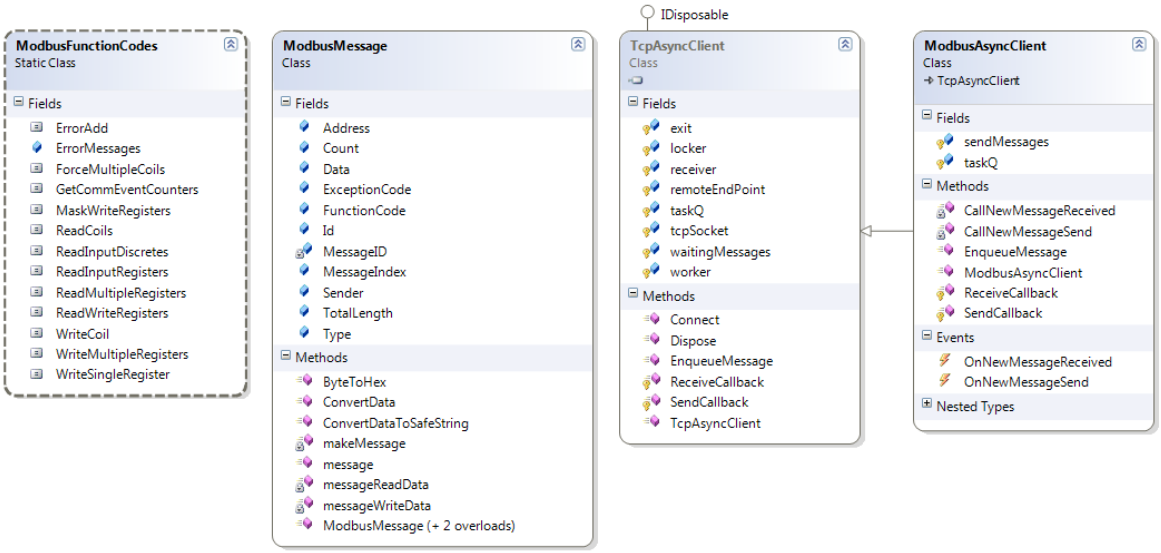
Celý systém byl navržen jako multiplatformní. Souběžně se systémem Damic je vyvíjena verze pro domovní automatizaci s PLC Wago, popsaná v diplomové práci Ondřeje Nývlta (16). Tento systém je založený na centrální řídicí jednotce, ke které jsou připojeny veškeré vypínače, světla, topení, ventilátory, atd. Systém je propojen s PC prostřednictvím Ethernetové sítě a protokolu Modbus. Klient pro Modbus byl rovněž vytvořen v rámci diplomové práce. Tento klient je schopen stáhnout kompletní konfiguraci z PLC Wago, vytvořit z ní příslušný model a ten nahrát na server. Pozměněný model opět převede a nahraje do PLC.

Modbus klient

Modbus je otevřený protokol pracující na různých fyzických sběrnicích. Umožňuje přenášení dat mezi různými zařízeními jako PLC, vstupní nebo výstupní zařízení a jiné. Komunikace funguje na principu klient - server. Protokol Modbus je implementován podle manuálu pro Wago (17). Obrázek 11.4 zobrazuje popis tříd, které zprostředkovávají komunikaci pomocí protokolu Modbus. Třída „*ModbusFunctionCodes*“ obsahuje konstanty definující čísla funkcí pro jednotlivé Modbusové zprávy. Třída „*ModbusMessage*“ vytváří na základě daných parametrů konkrétní zprávu v daném formátu pro Modbus a obráceně - přijatou zprávu

dekóduje a do příslušných proměnných doplní veškeré parametry. Třída „*TcpAsyncClient*“ je asynchronní klient pro protokol TCP. Instance dané třídy automaticky udržuje spojení, a v případě, že uživatel zařadí do fronty zprávu, odešle ji, jakmile je linka volná.

Třída „*ModbusAsyncClient*“ rozšiřuje třídu „*TcpAsyncClient*“ o práci s protokolem Modbus. Každá zpráva, která je odeslána, zůstává ve frontě, než na ni přijde odpověď. Ke každé přijaté odpovědi se přiřadí původní zpráva. Při každém přijetí zprávy nastane událost „*OnNewMessageReceived*“.

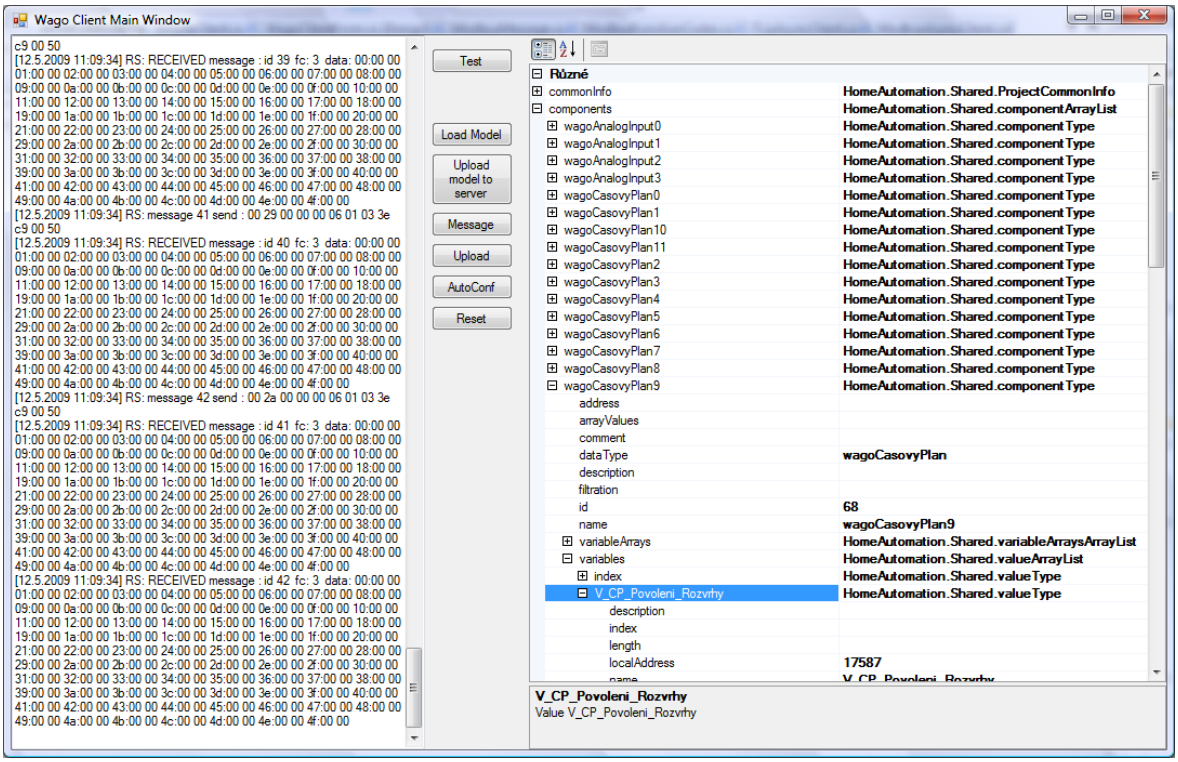


Obrázek 11.4: Popis tříd pro komunikaci prostřednictvím protokolu Modbus

WAGO Driver

Třída „*WagoBasicClient*“, následovník třídy „*SimpleClient*“, využívá instanci třídy „*ModbusAsyncClient*“ pro komunikaci s PLC Wago přes protokol Modbus. Modbus umožňuje přistupovat k částem paměti PLC mapovaných na Modbus, ale neumožňuje k nim přistupovat jako k objektům. Paměťové adresy jsou načteny z XML modelu, který obsahuje pouze definice typů virtuálních zařízení. Podle nich se vytvářejí požadavky na čtení paměti z PLC. Virtuálním zařízením se rozumí například analogový vstup, včetně všech jeho parametrů. V každé doručené zprávě je přenesena část paměti, která odpovídá různým datovým typům. Tyto zprávy jsou analyzovány podle své adresy a zapsány do příslušných proměnných v modelu. Takto je pro každý typ virtuálního zařízení vytvořeno specifické zpracování zpráv (paměti). Metoda „*DownloadModel*“ třídy „*WagoBasicClient*“ stáhne kompletní obraz paměti z Waga a

vrátí odkaz na vytvořený model. Naopak metoda „*UploadModel*“ vytvoří z modelu obraz paměti a ten pošle po Modbusu do PLC.



Obrázek 11.5: Testovací klient pro Wago

Testovací klient pro WAGO

Z časových důvodů byla pro Wago vytvořena pouze testovací verze klienta. Ten používá instanci třídy „*WagoBasicClient*“ pro načtení modelu. Obrázek 11.5 popisuje zmíněného testovacího klienta. V levé části jsou jednotlivé zprávy, přenášené přes Modbus, a v pravé části načtený model. Ve středním panelu jsou tlačítka umožňující načtení modelu, nahrání pozměněného modelu z klienta zpět do PLC, reset PLC a test jednotlivých funkcí. Model v pravé části lze libovolně procházet a editovat. V budoucnu bude pro Wago implementována podobná funkcionalita jako pro systém Damic.

11.5 Grafické rozhraní WPF

Téměř každá třída reprezentující grafiku ve WPF se skládá ze dvou částí: první představuje grafiku v XAML, druhá část popisuje logickou funkcionalitu v C#. Přímo v XAML lze vytvářet animace, měnit parametry vlastností v reakci na jednotlivé události, atd. Pomocí deklarace v XAML je možné vytvořit instance jednotlivých tříd,

které jsou v kódu pod stejným jménem jako je hodnota atributu „*Name*“. Ukázka jednoduché stránky v XAML je uvedena níže.

```
<Page      Element Page specifikuje třídu, od které se dědí, atribut x:Class specifikuje konkrétní jméno třídy
    x:Class="HomeAutomation.Clients.Gui.Wpf.WelcomePage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:my="clr-namespace:HomeAutomation.Clients.Gui.Wpf.Components;
              assembly=WpfPageComponents"
    .
    .
    .
    x:Name="Window"
    Title="Okno"
    Width="1024"
    Height="768">
    <Grid>
       .
       .
       .
       <my:Header  PanelText="" Name="Nadpis"/>
       <my:BigIcon IconText="Vizualizace"
        MouseDown="BigIcon_Vizualizace_MouseDown"
        ButtonImage="/GUIVisualisationClientWPF;
              component/images/zeus-visualizace.png" />
    </Grid>
</Page>
```

Atributem xmlns se specifikuje použité XML schéma. Pokud se používají komponenty specifikované v jiném namespace, je nutné za atribut xmlns uvést identifikátor, kterým se bude do namespace přistupovat, zde „my“. Hodnota atributu je v tomto případě namespace a sestavení podle příkladu.

Instance tříd z namespace definovaného identifikátorem „my“. Veřejné vlastnosti nebo metody jednotlivých instancí lze uvést jako atributy. V kódu se jednotlivé instance jmenují stejně jako v XAML hodnota parametru „Name“.

11.5.1 Grafické komponenty

V následující kapitole je uveden seznam všech grafických komponent použitých v GUI.

iqHomeHeater

Komponenta reprezentuje stav topení ve vizualizaci. V případě, že je topení zapnuté, objeví se okolo komponenty zář.

Obrázek 11.6: Grafická komponenta iqHomeHeater

iqHomeLight

Komponenta má téměř stejnou funkčnost jako komponenta topení. Navíc obsahuje roletové menu pro ovládání světel.

Obrázek 11.7: Grafická komponenta iqHomeLight

ButtonTouchPart

Třída reprezentující jedno tlačítko na vypínači ve vizualizaci. Obsahuje roletové menu se seznamem možných akcí, které tlačítko může provést.

Obrázek 11.8: Grafická komponenta ButtonTouchPart

ButtonTouch

Tato třída slouží jako kontejner pro jednotlivá tlačítka „*ButtonTouchPart*“ ve vizualizaci. Komponenta standardně obsahuje jedno, dvě nebo čtyři tlačítka.

Obrázek 11.9: Grafická komponenta ButtonTouch

DayProgramEditLite

Pro nastavení denního teplotního plánu se používá komponenta „*DayProgramEditLite*“. Umožňuje zvolit jednu až tři změny a nastavit výšku a šířku jednotlivých hrbů.

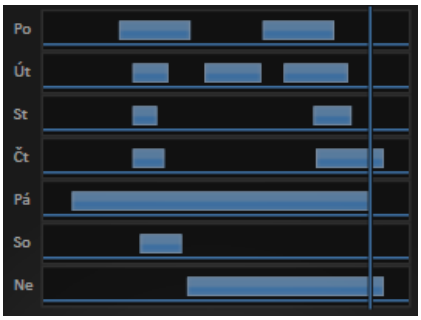
Obrázek 11.10: Grafická komponenta DayProgramEditLite

TreeControl

Komponenta odvozená ze třídy „*TreeView*“. Umožňuje editovat stromovou strukturu a myší přetahovat jednotlivé komponenty.

ChooseMode

Pro přehled nastaveného plánu pro celý týden je vytvořena komponenta ChooseMode. Komponenta je seznamem komponent „*DayBox*“, které zobrazují plán pro konkrétní den.



Obrázek 11.11: Grafická komponenta ChooseMode

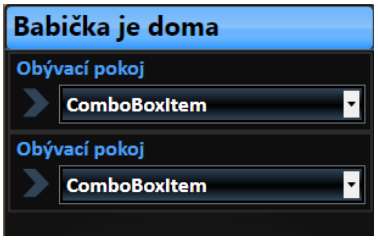
RoomBox

Pro volbu schématu a režimu na daném termostatu slouží komponenta „*RoomBox*“.

Obrázek 11.12: Grafická komponenta RoomBox

RoomPanel

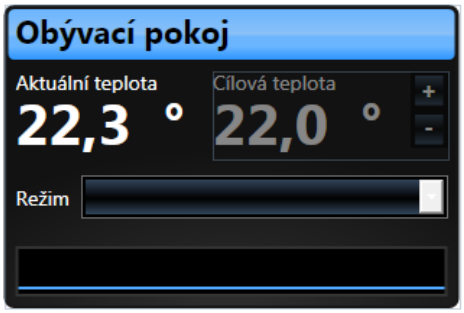
Nastavení režimu pro celý dům je možné pomocí komponenty „*RoomPanel*“. Pomocí rolovacího menu lze vybrat pro každý pokoj vybrané schéma.



Obrázek 11.13: Grafická komponenta RoomPanel

TempWidget

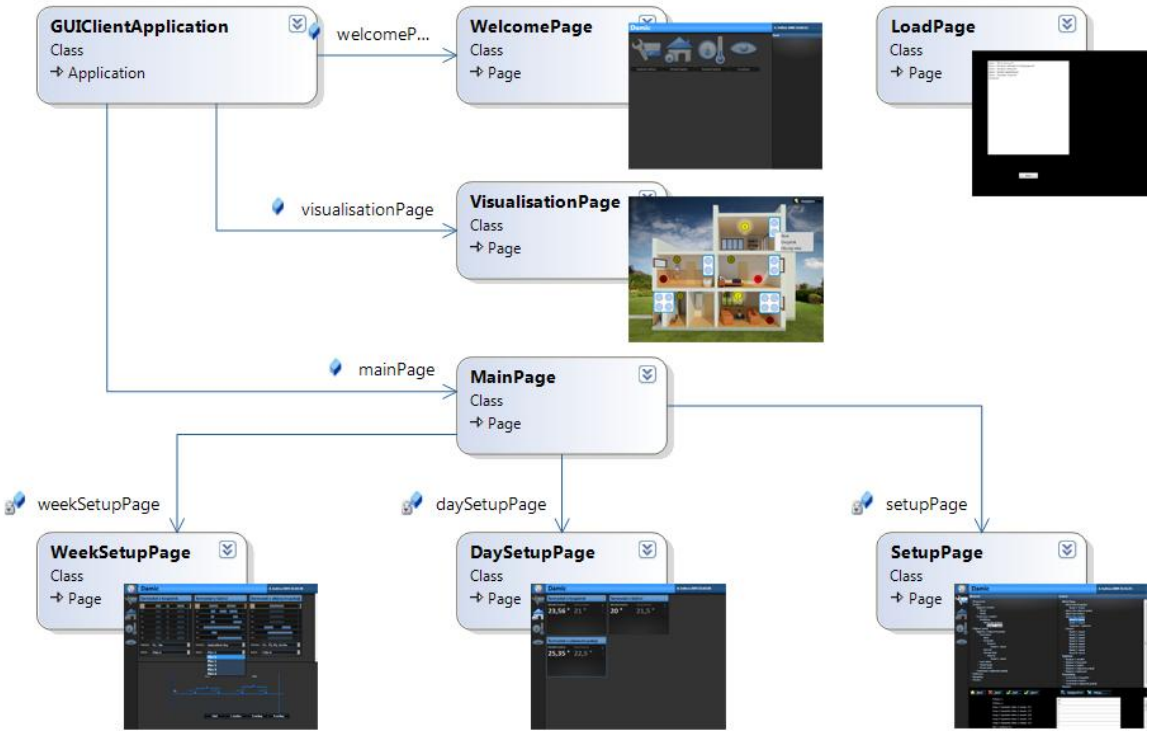
Přehled a nastavení aktuální teploty se realizuje prostřednictvím komponenty „*TempWidget*“.



Obrázek 11.14: Grafická komponenta TempWidget

11.5.2 Grafický klient

Grafický klient (sestavení „*GUIVisualisationClientWpf*“) využívá pro komunikaci se serverem třídu „*GUIClient*“ a pro zobrazení grafických prvků všechny grafické komponenty uvedené v minulé kapitole. Pro každou obrazovku je vytvořena vlastní třída, založená na třídě „*Page*“. Obrázek 11.15 popisuje strukturu jednotlivých stránek. Aplikace „*GUIClientApplication*“ inicializuje komunikaci se serverem. V případě, že model není dostupný a uživatel žádá načtení modelu ze serveru, přejde na stránku „*LoadPage*“. Z této stránky je uživatel po zvolení souboru automaticky přesměrován na „*WelcomePage*“. Mezi stránkami „*WelcomePage*“, „*VisualisationPage*“ a „*MainPage*“ lze libovolně přecházet prostřednictvím referencí v „*GUIClientAppliaction*“. Stránka „*MainPage*“ je pouze obálkou pro stránky se specifickou funkcionalitou, jako jsou „*SetupPage*“, „*DaySetupPage*“, „*WeekSetupPage*“, které se vkládají do této stránky.



Obrázek 11.15: Souhrn jednotlivých obrazovek aplikace

12 ZÁVĚR

Práce obsahuje zcela funkční vizualizační a konfigurační nástroj pro systém domovní automatizace Damic a zároveň rozpracovává implementaci pro systém s PLC Wago. Aplikace byla navržena tak, aby v nejvyšší možné míře splnila požadavek univerzálnosti celého řešení, a zároveň neomezovala aspekty, které jsou v rozporu s tímto požadavkem. V průběhu testování na modelu domu bylo ověřeno, že uživatel je schopen pomocí vytvořené aplikace plně nakonfigurovat reálný systém. Některé navržené způsoby nastavení, především vytváření vazeb mezi jednotlivými zařízeními pomocí stromové struktury, se ukázaly jako příliš komplikované a uživatelsky nevhodné. V následujících verzích se počítá s jiným, jednodušším způsobem tvorby vazeb.

Navržená architektura systému splňuje všechny požadavky uvedené na začátku této práce. Pro plné využití je nutné vyřešit problém komunikace dvou počítačů v rozdílných sítích, skrytých za NATem. Aplikace splňuje základní část úvodních požadavků; realizace ostatních, jako například návrh různých úrovní oprávnění přístupu k modelu, další funkce systému Damic, alarmy, diagnostika, dynamická konfigurace, správa většího počtu budov atd., je plánována do následujících verzí aplikace.

Navržený model se při praktických testech osvědčil a omezení bylo zaznamenáno pouze u zařízení s velkým počtem parametrů, jako je například termostat s více časovými plány. Omezení se projevilo pouze tím, že byla změněna struktura uložených plánů. Model je popsán pomocí otevřeného formátu, znamená to tedy, že umožňuje použití jinými nástroji.

BIBLIOGRAFIE

1. XML intro. *PLCOpen.* [Online] PLCOpen. [Citace: 23. 4 2009.] http://www.plcopen.org/pages/tc6_xml/xml_intro/index.htm. .

2. **MacDonald, Matthew.** *Pro WPF in C# 2008.* New York : Apres, 2008. ISBN 1-4302-0576-8.

3. **McLean, Scott, Naftel, James a Williams, Kim.** *Microsoft .NET Remoting.* Washington : Microsoft Press, 2003. Part No. X08-81840.

4. **Rammer, Ingo a Szpuszta, Mario.** *Advanced .NET Remoting Second Edition.* New York : Apres, 2005. str. 593. ISBN 1-59059-417-7.

5. Damic - uACT 010. *Midam Control System.* [Online] 2009. [Citace: 29. 4 2009.] http://www.midam.cz/dnld/uact010/cz/Katalogov%FD%20list.pdf.

6. Damic - uDIM 010. *Midam Control System.* [Online] 2009. [Citace: 29. 4 2009.] http://www.midam.cz/dnld/udim010/cz/Katalogov%FD%20list.pdf.

7. Damic - uLMI 010. *Midam Control System.* [Online] 2009. [Citace: 29. 4 2009.] http://www.midam.cz/dnld/ulmi010/cz/Katalogov%FD%20list.pdf.

8. Damic - uLMO 010. *Midam Control System.* [Online] 2009. [Citace: 29. 4 2009.] http://www.midam.cz/dnld/ulmo010/cz/Katalogov%FD%20list.pdf.

9. Damic - uLSW 010. *Midam Control System.* [Online] 2009. [Citace: 29. 4 2009.] http://www.midam.cz/dnld/ulsw010/cz/Katalogov%FD%20list.pdf.

10. Damic - uLTH 010. *Midam Control System.* [Online] 2009. [Citace: 29. 4 2009.] http://www.midam.cz/dnld/ulth010/cz/Katalogov%FD%20list.pdf.

11. Damic. *Midam Control System.* [Online] Midam, 2009. [Citace: 29. 4 2009.] http://www.midam.cz/index.php?id=200&cat=9.

12. **Píša, Pavel.** What is uLan? *uLan communication.* [Online] 2009. [Citace: 30. 4 2009.] http://ulan.sourceforge.net/.

13. —. ul_drv - uLan RS-485 Communication Driver. *uLan.* [Online] 2009. [Citace: 30. 4 2009.] http://cmp.felk.cvut.cz/~pisa/ulan/ul_drv.html.

14. **Corby, Karen.** VS Template: Flexible Application. *Scorbs.* [Online] 4. 6 2006. [Citace: 15. 2 2009.] http://scorbs.com/2006/06/04/vs-template-flexible-application.

15. **Sells, Chris a Griffiths, Ian.** *Programming WPF.* Sebastopol : O'Reilly, 2007. ISBN-10: 0-596-51037-3.

16. **Nývlt, Ondřej.** *Automatický návrh řízení pro domovní automatizaci - Diplomová práce.* Praha : ČVUT Praha, 2008.

17. **WAGO Kontakttechnik GmbH.** *WAGO IO System 750, Modular IO System, Ethernet TCP/IP, Manual.* Minden : WAGO Kontakttechnik GmbH, 2001.

18. What Is Windows Communication Foundation? *MSDN.* [Online] Microsoft, 2007. [Citace: 26. 4 2009.] http://msdn.microsoft.com/en-us/library/ms731082.aspx.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

.NET	Framework pro systém Windows
API	Application Programming Interface; rozhraní pro přístup ke knihovnám, systémovým službám
COM	Component Object Model; technologie meziprocesní komunikace
DCOM	Distributed COM; technologie meziprocesní komunikce v síťovém prostředí
DirectX	Soubor API pro přístup k multimédiím, například hardwaru grafiké karty
DLL	Dynamically Linked Library; knihova připojovaná za běhu programu
Firewall	Program blokující neautorizovaný přístup do počítače
Framework	softwarová struktura sloužící pro podporu programování
GUI	Graphical User Inteface; uživatelské rozhraní umožňující interakci pomocí grafických prvků
HTML	HyperText Markup Language; značkovací jazyk vytvořený pro webové stránky
NAT	Network Address Translation; překlad síťové adresy v paketu na jinou (často veřejnou)
PLC	Programmable Logic Controller; programovatelný logický automat
Port forwarding	Mapování portu počítače za NATem na port routeru, který provádí NAT
Proxy	Síťová služba umožňující nepřímé spojení se vzdálenou službou
RS485	Specifiakce dvoudrátového multibodového sériového spoje
Serializace	Seřazení objektu do určitého formátu
SOAP	Simple Object Access Protocol; protokol pro výměnu dat v počítačových sítích
TCP	Transmission Control Protocol; jeden ze základních protokolů internetu
UART	Universal Asynchronous Receiver/Transmitter; hardware, který překládá mezi paralelní a sériovou komunikací
uLAN	Komunikační protokol pro linku RS-485
URL	Uniform Resource Locator; řetězec znaků specifikující přesné umístění dat v internetu
WCF	Windows Communication Foundation; framework pro meziprocesní komunikaci
WPF	Windows Presentation Foundation; grafický framework umožňující vytvářet graficky bohaté prvky
XAML	Extensible Application Markup Language; jazyk pro popis grafických prvků ve WPF
XBAP	XAML Browser Applications; WPF aplikace spustitelné v internetovém prohlížeči
XML	Extensible Markup Language; obecný značkovací jazyk

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

NÁVOD NA SPUŠTĚNÍ DEMO APLIKACE NA CD/DVD

Aplikace je umístěna v adresáři „Demo aplikace“. Je možné ji spustit pomocí automatické nabídky na CD – program „DamicStart.exe“ nebo přímo pomocí skriptu „startDamic.bat“ v daném adresáři. Skript je nutné spustit s administrátorskými právy. Celá aplikace vyžaduje pro korektní běh .NET Framework 3.5 Service Pack 1 (verze 3.5.30729). Verzi nainstalovaného frameworku na počítači lze zjistit v automatické nabídce na CD.

Skript po spuštění nainstaluje službu serveru a spustí kontrolní aplikaci serveru. Poté se ptá na jednotlivé aplikace. Tyto aplikace se spustí zadáním klávesy „a“ (ano), případně nespustí „n“ (ne), a potvrzením klávesou Enter. Grafické vizualizační a konfigurační aplikace jsou „Damic GUI a Damic GUI XBAP“. Editor modelu je možné použít na náhled a procházení modelu. Damic driver je pouze na ukázku, protože bez připojeného hardwaru nemá žádnou funkci. Damic hardware simulator je demonstrační aplikace, která náhodně mění stavové hodnoty v modelu tak, aby simulovala chování obyvatelů domu. Každou změnu hodnoty vypíše do své stavové konzoly. Jednotlivé změny je možné pozorovat pomocí Damic GUI (XBAP).

Pro ukončení všech aplikací a odinstalování serveru stačí potvrdit tuto volbu ve skriptu a vše se vykoná automaticky.

Adresář obsahuje následující podadresáře s jednotlivými programy:

1. Server service
2. Server service control form
3. Damic gui
4. Damic gui XBAP
5. Editor modelu
6. Damic driver
7. Damic hardware simulator

Pokud je potřeba službu „Server service“ instalovat manuálně, jsou zde opět připravené skripty „install.bat“ a „uninstall.bat“. Oba musí být spuštěny s administrátorskými právy.

OBSAH CD/DVD

Médium obsahuje kromě výše popsané Demo aplikace tuto diplomovou práci a veškeré zdrojové kódy. Zdrojové kódy jsou v adresáři „Zdrojove soubory“. Adresář „Template pro Visual Studio“ obsahuje soubor „Flexible Application (WPF).zip“. Tento template je popsán v kapitole 9. Instalace se provede tak, že zabalený soubor nakopírujeme do adresáře:

„%userprofile%\Documents\Visual Studio 2008\Templates\ProjectTemplates\Visual C#\“.

České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídicí techniky

Stručný úvod do konfigurace systému Damic

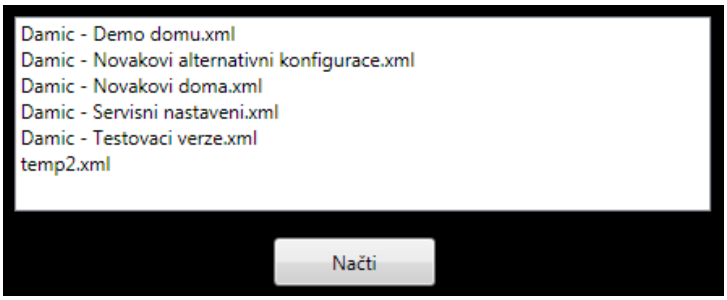
Uživatelský manuál

Bc. Ivo Kubita

Praha 2009

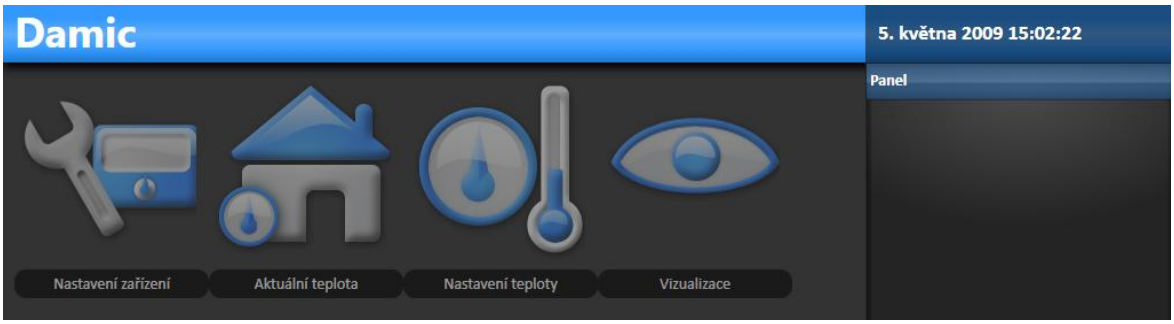
OBSAH

1 SPUŠTĚNÍ APLIKACE



Obrázek 1.1: Načtení modelů ze serveru

Aplikace se při spuštění připojí k serveru, zkontroluje, jestli je na serveru umístěný model. V kladném případě s ním začne pracovat. Pokud zjistí, že na serveru žádný model není, nabídne uživateli volbu, jestli načíst model z lokálního souboru nebo ze souboru umístěného na serveru. Volba lokálního modelu je standardní okno pro otevření souboru. Pokud se uživatel rozhodne pro model uložený na serveru, aplikace stáhne seznam všech souborů, které server nabízí, a zobrazí nabídku uživateli, viz Obrázek 1.1. Uživatel vybere jeden model a aplikace sdělí serveru, který model má nahrát. Server model načte ze souboru a poskytne k němu přístup všem klientům. Po připojení k načtenému modelu, přejde aplikace na úvodní obrazovku, viz Obrázek 1.2. Na obrazovce najdeme čtyři ikony, kterými můžeme zvolit, co chceme dále se systémem dělat. První ikona nás nasměruje ke kompletní konfiguraci systému. Kliknutím na druhou se dostaneme do nabídky všech termostatů a můžeme změnit požadovanou aktuální teplotu. Třetí ikona nás zavede do nastavení teplotních křivek pro každý termostat a za poslední ikonou se skrývá vizualizace světel, topných těles a spínačů.



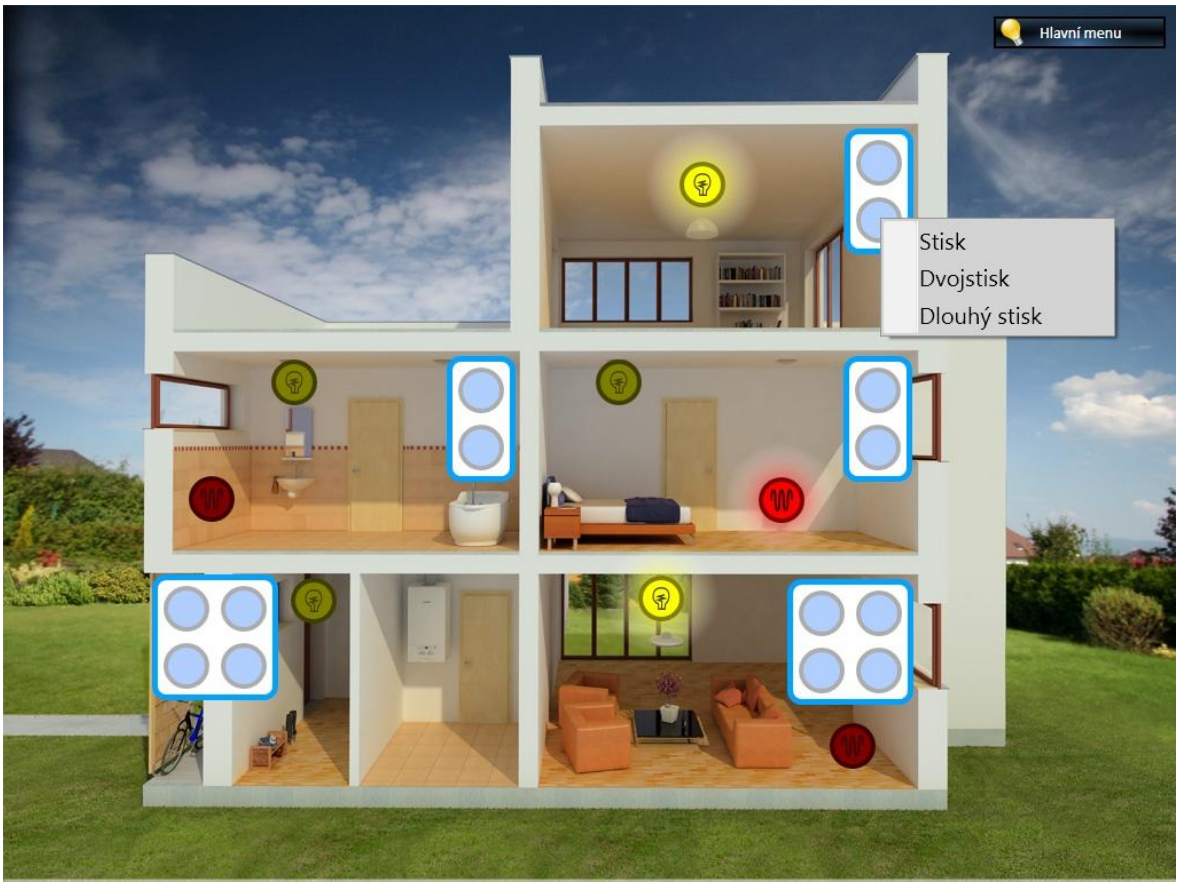
Obrázek 1.2 Úvodní obrazovka

2 VIZUALIZACE

Vizualizace slouží k přehledu stavů všech světel, topných těles a jejich rychlému ovládání. Samozřejmostí je simulace různých druhů stisků jednotlivých tlačítek. Obrázek 2.2 popisuje vzorovou vizualizaci prezentačního panelu. Na panelu jsou umístěna čtyři světla připojená na stmívač, jedno světlo připojené na výstup akčního členu, tři topná tělesa připojená na výstupy tří jednoduchých akčních členů, tři dvoutlačítkové a dva čtyřtlačítkové spínače. Světla jsou znázorněna žlutozelenými kolečky. Pokud světlo svítí s jakoukoli intenzitou, je okolo něj žlutá poloprůhledná zář. Podobně jsou zobrazena topná tělesa – červená kolečka. Ovládání jednotlivých světel je řešeno kliknutím na ikonu požadovaného světla: objeví se kontextové menu s nabídkou možných akcí. Obrázek 2.1 popisuje rozdíl mezi ovládáním světla připojeného ke stmívači a k akčnímu členu. Světlo připojené k akčnímu členu (obrázek vlevo) můžeme pouze rozsvítit, zhasnout nebo negovat aktuální stav. Světlo připojené na stmívač podporuje řadu funkcí, nicméně ve vizualizaci jsme se rozhodli použít zatím pouze tři základní – zhasnutí, rozsvícení na maximum a rozsvícení na poslední zadanou hodnotu. Jak nastavit požadovanou hodnotu jasu, je popsáno v kapitole Konfigurace systému. Topení nelze z vizualizace ovládat přímo, ale pouze pomocí nastavení termostatů, které je popsáno v kapitole Zobrazení a nastavení aktuálních teplot.

Obrázek 2.1: Vizualizace ovládání světel, vlevo jednoduché světlo připojené k akčnímu členu, vpravo světlo připojené na stmívač

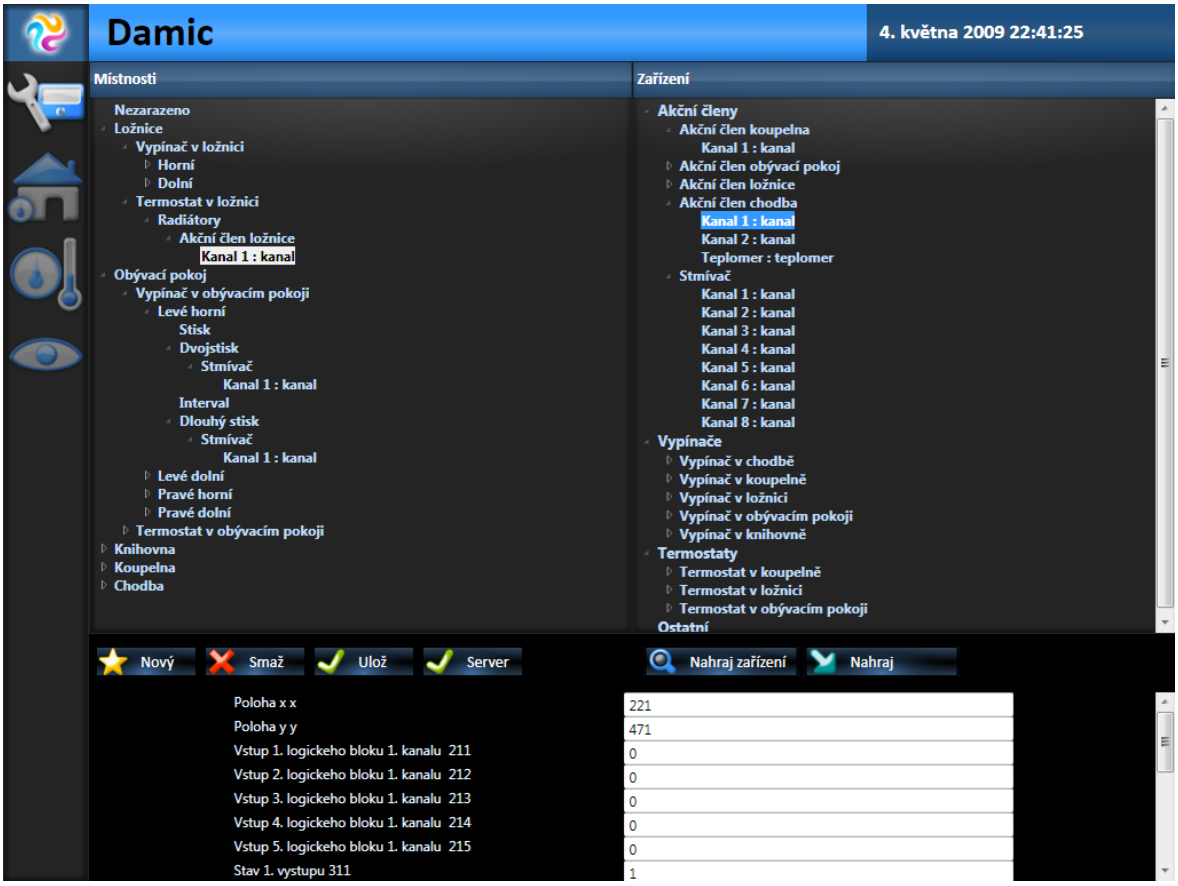
Vypínače jsou reprezentovány modrými obdélníky s vyznačenými stisknutelnými plochami. Kliknutím na vypínač se nám zobrazí podobné kontextové menu jako u světel. Zobrazené akce stisk, dvojstisk a dlouhý stisk simulují skutečnou akci na vypínači. Volbou požadované akce vykoná vypínač stejnou funkci, jako kdyby bylo stisknuto příslušné tlačítko.



Obrázek 2.2: Ukázka vizualizace

3 KONFIGURACE SYSTÉMU

Systém se konfiguruje pomocí dvou stromových struktur. Obrázek 3.1 popisuje konfigurační okno. V levé části jsou jednotlivé místnosti (filtrační kategorie), v pravé jsou jednotlivé komponenty systému roztříděné podle příslušných kategorií. Ve spodní části se nastavují detailní parametry zařízení nebo vazeb. Ikony na levé straně slouží k rychlému přepnutí do vizualizace nebo nastavení jiných částí systému.

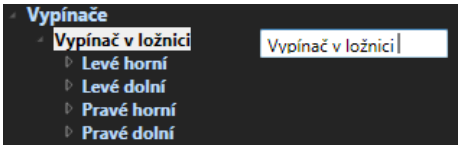


Obrázek 3.1: Konfigurace systému

Ikony pod stromovými strukturami mají následující význam:

- Levý panel
  - Nový – vytvoří novou místnost (filtrační kategorii)
  - Smaž – smaže vybranou místnost nebo odebere vybrané zařízení z místnosti
  - Ulož – zobrazí dialogové okno pro uložení modelu lokálně
  - Server – zobrazí dialogové okno pro uložení modelu na server
- Pravý panel
  - Nahraj zařízení – nahraje vybrané zařízení do hardwaru
  - Nahraj – nahraje celý model do hardwaru

Všechna zařízení se dají libovolně přejmenovat. Dvojklikem na požadovaném zařízení zobrazíme editační okénko, kde lze název přepsat - viz Obrázek 3.2.



Obrázek 3.2: Přejmenování komponenty

3.1 Přiřazení vypínačů do místností

Pokud je konfigurován systém, který je kompletně načtený z hardwaru, není dostupná žádná informace o místnostech a všechna zařízení budou v kategorii „Nezařazeno“. V tomto případě je potřeba v levém panelu vytvořit požadované místnosti. Do těchto místností můžeme z pravého panelu přesunout příslušné vypínače. Každý vypínač lze rozbalit; podřazené položky značí jednotlivá tlačítka na vypínači – například levé horní, levé dolní, atd. – viz Obrázek 3.2. Každé z těchto tlačítek má v podřazených položkách všechny akce (stisk, dvojstisk, dlouhý stisk, interval – delší než dlouhý stisk), kterým je možné přiřadit nějakou reakci. Obrázek 3.3 zobrazuje rozbalenou stromovou strukturu.

Pokud je vybráno zařízení v levém panelu, zobrazí se ve spodním panelu seznam všech možných nastavení, viz Obrázek 3.1. Jakýkoli z těchto parametrů lze nastavit na požadovanou hodnotu.

3.2 Konfigurace světel

Světlo může být fyzicky připojeno buď na stmívač, nebo na akční člen. Vazba mezi světlem a vypínačem se provádí přetažením požadovaného kanálu stmívače nebo akčního členu na požadovanou akci vypínače. Přetahuje se pouze jeden kanál zvoleného zařízení, nikoli celé zařízení. Příklad: Vypínačem v obývacím pokoji chceme naplno rozsvítit světlo taktéž v obývacím pokoji, které je připojené na první kanál stmívače. Obrázek 3.3 popisuje výslednou konfiguraci. Vybereme si tedy některé tlačítko vypínače, v našem případě levé horní. V pravém panelu vybereme první kanál stmívače a myší přesuneme na zvolenou akci levého horního tlačítka vypínače v obývacím pokoji. Stmívač s jedním kanálem se vloží do levého stromu. Pokud klikneme na stmívač v levém stromě, v dolním panelu se zobrazí výběr akce, která se má provést se světlem po dvojstisku tlačítka - vyberme „rampa nahoru“. Zde se dá

vybrat ze všech možností, které stmívač podporuje – viz Obrázek 3.3. Každé akci na vypínači lze přiřadit až pět reakcí různých světel. Takto je nakonfigurována vazba mezi světlem a vypínačem v modelu. Aby byla tato funkce uvedena v činnost, musí se model nahrát do hardwaru. Nahrát lze buď celou konfiguraci, nebo jen jedno zařízení. Kliknutím na „Nahraj“ se nahraje celý model. Pokud byla změna provedena pouze v jednom zařízení, stačí je vybrat v pravém seznamu a zvolit „Nahraj zařízení“. V našem případě bychom vybrali vypínač v obývacím pokoji. Po nahrání vypínač při dvojstisku pomalu naplno rozsvítí světlo připojené na první kanál stmívače. Stejným způsobem se nakonfiguruje i světlo připojené na akční člen. Rozdíl je pouze v dostupných reakcích – nebude moci rozsvěcovat a zhasínat rampu, ale naopak bude moci, například, negovat aktuální stav světla.

Obrázek 3.3: Konfigurace funkce vypínače

3.3 Konfigurace topných těles

Termohlavice topných těles se fyzicky připojují na výstup akčního členu. Topná tělesa jsou spínána termostatem. Tato vazba se opět provádí obdobným způsobem jako u

světel, pouze s několika málo rozdíly. Termostat obsahuje přímo podřazenou položku „Topení“. Do této položky se přímo přetáhne kanál akčního členu připojeného na termohlavici. Nahráním konfigurace termostatu do hardwaru začne tato vazba fungovat.

4 ZOBRAZENÍ A NASTAVENÍ AKTUÁLNÍCH TEPLOT

Kliknutím na ikonu „Aktuální teplota“ (druhá odshora v menu vlevo) se dostaneme do přehledu všech termostatů, viz Obrázek 4.1. Panel každého termostatu zobrazuje vlevo aktuální teplotu a vpravo cílovou teplotu z daného aktuálního topného programu. Tlačítky „+“ a „-“ lze cílovou teplotu dočasně měnit. Platnost takto změněné teploty je půl hodiny od poslední změny, poté vejde v platnost opět časový plán.

Obrázek 4.1: Zobrazení a nastavení teplot na termostatech

5 NASTAVENÍ ČASOVÝCH PLÁNŮ

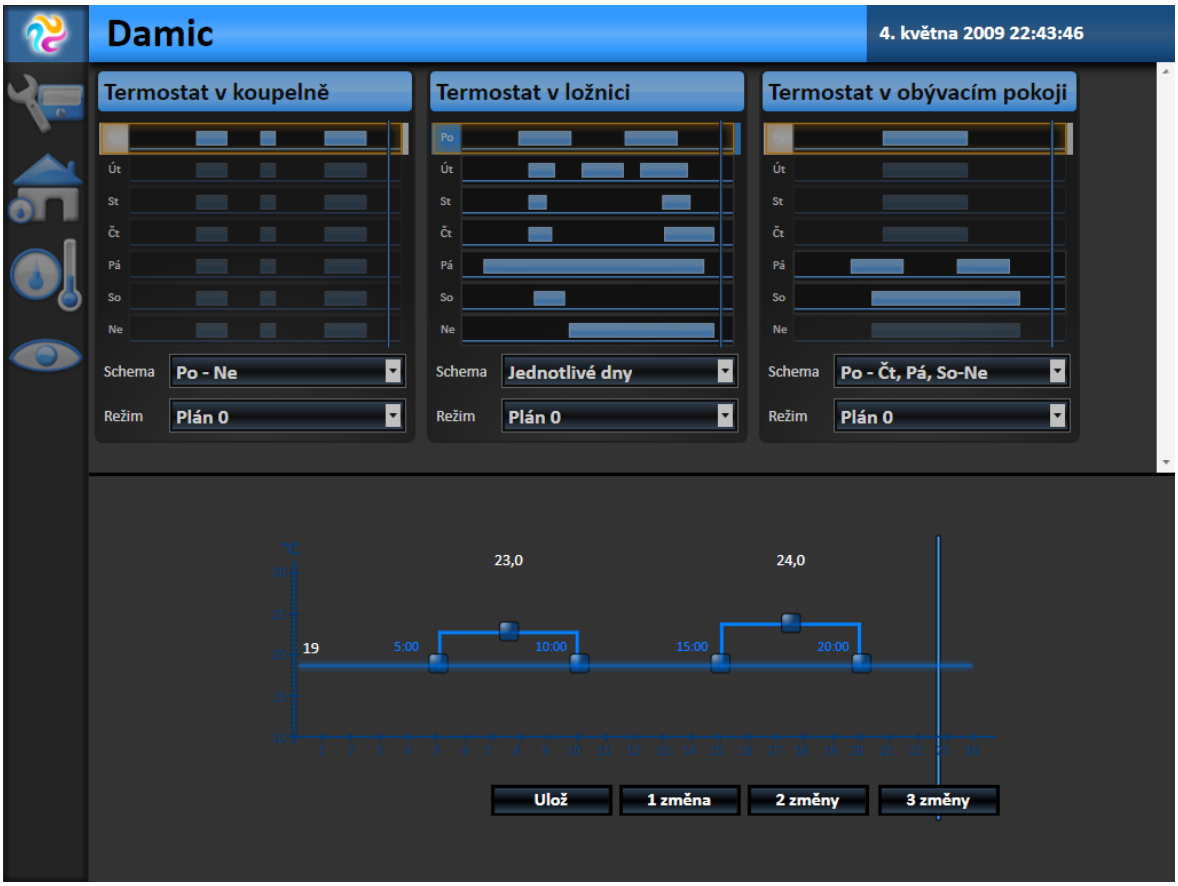
Každému termostatu lze nastavit pět rozdílných časových plánů a kdykoli je libovolně přepnout. Nastavení je možné jak na PC, tak i přímo na termostatu. Nastavovací okno se dělí na dvě hlavní části, viz Obrázek 5.2. V horní části je přehled všech termostatů s náhledem týdenního časového plánu, volbou schématu a volbou plánu. U každého plánu lze nastavit čtyři různá schémata:

- Jednotlivé dny:každý den se nastavuje zvlášť.
- Pondělí – pátek, sobota – neděle: pracovní týden a víkend se nastavují zvlášť.
- Pondělí – čtvrtek, pátek, sobota – neděle: pracovní týden kromě pátku, pátek a víkend se nastavují zvlášť.
- Pondělí – neděle: celý týden stejný časový plán, nastavuje se pouze jeden den.

Podle vybraného schématu se do následujících dnů kopíruje nastavení. Obrázek 5.1 znázorňuje volbu režimu a schématu.

Obrázek 5.1: Volba režimu a schématu na termostatu

Vybranou teplotní křivku lze editovat ve spodní části okna. Editace teplotní křivky spočívá v nastavení základní teploty, která je pro celý časový plán stejná, a ve volbě jedné až tří změn denně. Volba těchto změn se provádí příslušnými tlačítky ve spodní části panelu. Teplota se dá nastavit jednoduchým přetažením modrého čtverečku, umístěného uprostřed grafu změny, vertikálním směrem, stejně tak čas příslušné změny se nastavuje přetažením modrého čtverečku na začátku, popřípadě na konci změny, v tomto případě horizontálním směrem. Čas lze nastavit v rozsahu čtvrthodin a teplotu v rozsahu půl stupně. Poté, co průběh teploty získá požadovaný tvar, je potřeba potvrdit zápis do modelu tlačítkem „Ulož“. Změny se projeví až po nahrání konfigurace do hardwaru (jednoho změněného zařízení nebo kompletní konfigurace).



Obrázek 5.2: Nastavení časových plánů