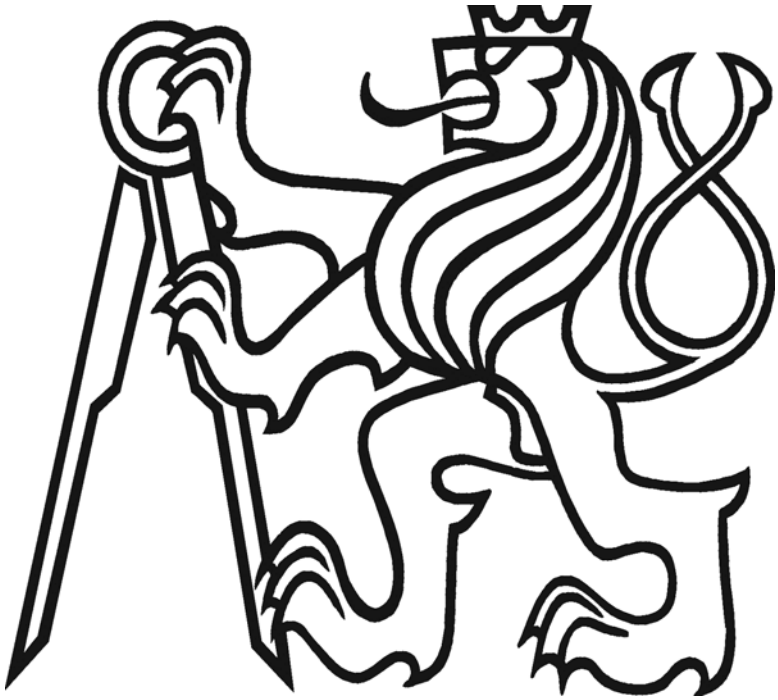
České vysoké učení technické
Fakulta elektrotechnická
Katedra řídící techniky



Implementace SW pro řízení inteligentní elektroinstalace

Bakalářská práce

Vedoucí: Ing. Pavel Němeček

Autor: Josef Mrázik

Praha 2006

Prohlášení

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

V Praze dne …….. ……..
Podpis

Anotace

Cílem práce je navrhnout strukturu a následně implementovat konfigurační software pro systém inteligentní elektroinstalace tak, aby bylo pomocí tohoto softwaru možno definovat veškeré funkce jichž je uvažovaný systém schopen.Tyto funkce jsou definovány uživatelem skrze grafické rozhraní, jež je schopno předávat informace o konfiguraci také přes Internet, s využitím základních funkčních primitiv-podmínek,událostí a akcí. Uvažovaný systém inteligentní elektroinstalace je tvořen moduly s různou funkcionalitou na jejichž typu a množství musí být konfigurační systém nezávislý. Proto byl navržen pro práci s definičními soubory určujícími vlastnosti užívaných modulů. Pro použití jakéhokoliv modulu je tedy třeba pouze jeho definiční soubor, který je softwarem zpracován.

Annotation

The purpose of the work is to design and implement a configuration software for intelligent house wiring system, which contains functions for defining all functions possible in the specific system. These functions are set by the user through graphic interface by combining primitive functions (conditions, events and actions). The software is capable of sending new configuration to another computer through the Internet. The house wiring system under consideration consists of modules with specific functionality and the software has not to be dpendent on their types or ammount. To guarantee this independence, a structure which uses special files to store information about the module's specification, was used, so for adding and using a new type of module only it's definition file is neccesary.

Obsah

1 Popis použitých technologií

1.1 Značkovací jazyka XML

XML (*eXtensible Markup Language*, česky *rozšiřitelný značkovací jazyk*) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem *W3C (World Wide Web Consortium).* Umožňuje snadné vytváření konkrétních značkovací jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro výměnu dat mezi aplikacemi, publikování dokumentů a uchovávání dat aplikací.

Tento formát je otevřený,tedy zdarma přístupný pro všechny uživatele, a není úzce svázán s nějakou platformou nebo proprietární technologií, což v důsledku znamená, že je, v případě potřeby zpracovatelný libovolným textovým editorem. Filosofií jazyka je popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, přičemž se ovšem sám o sobě nezabývá vzhledem dokumentu nebo jeho součástí. Prezentaci dokumentu (vzhled) je možné definovat připojeným stylem. Další, s tímto související možností poskytovanou tímto formátem je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML.

Hlavní výhodou jazyka XML je, že nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než je u jeho předchůdců i konkurentů obvyklé. Pomocí XML značek (*tagů*) vyznačujeme v dokumentu význam jednotlivých částí textu. Dokumenty tak obsahují více informací, než kdyby se používalo značkovaní zaměřené na prezentaci (vzhled) – definice písma, odsazení a podobně. XML dokumenty jsou informačně bohatší. To lze samozřejmě s výhodou využít v mnoha oblastech. Největší přínos bude samozřejmě pro prohledávání, kdy můžeme určit i jaký význam má mít hledaný text.

Další výhodnou vlastností je fakt, že jako znaková sada se implicitně používá ISO 10646 (také *Unicode*). V XML proto můžeme snadno vytvářet dokumenty, které obsahují znaky národních abeced. Problémy s konverzí z jednoho kódování do druhého odpadají díky možnosti současně použít i jiné kódování dle konkrétních požadavků, ovšem takové kódování musí být v každém dokumentu přesně definováno.

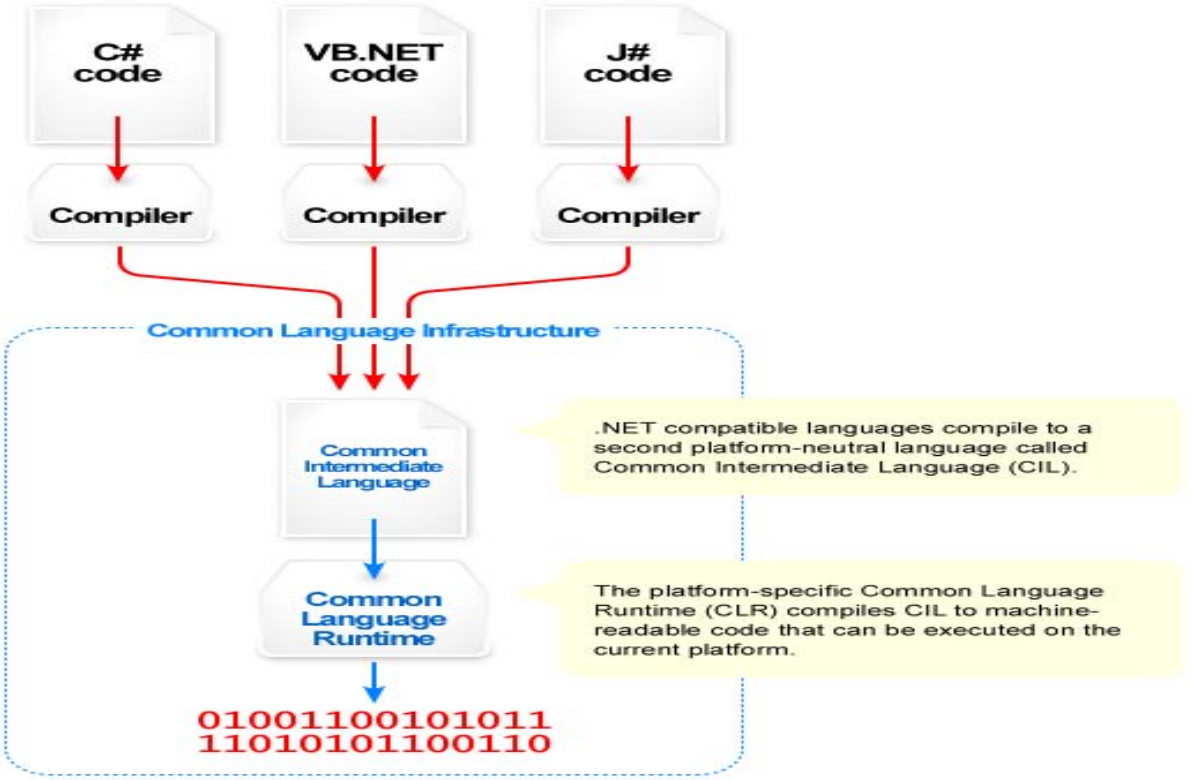
XML neobsahuje předdefinované značky ,je tedy třeba definovat v dokumentu vlastní značky, které budeme používat. Tyto značky je možné (nepovinně) definovat v souboru DTD (*Document Type Definition*). Potom je možné automaticky kontrolovat, zda vytvářený XML dokument odpovídá této definici. Tato kontrola je prováděna programem nazvaným parser. Při vývoji aplikací můžeme parser použít, a ten za nás detekuje většinu chyb v datech.

Další vlastností XML je, že v jednom dokumentu můžeme používat najednou nezávisle na sobě několik druhů značkovaní pomocí jmenných prostorů (namespaces). To umožňuje kombinovat v jednom dokumentů několik různých definic ve formě DTD nebo schémat bez konfliktů v pojmenování elementů.

1.2 Microsoft .NET Framework

Pod tímto názvem se skrývá komponenta operačního systému Microsoft Windows, poskytující množství již zpracovaných programových řešení pro běžně nastávající programové požadavky (například numerické algoritmy, uživatelská rozhraní, síťová komunikace) a obsahující běhové prostředí pro vykonávání programů napsaných pro toto prostředí jedním z podporovaných jazyků (základními jsou Visual Basic, J#, C# a C++) nazvané *Common Language Runtime*. Velkou výhodou je, že při dodržení jstých omezení lze přecházet od jednoho jazyka k druhému bez nutnosti manuálně upravovat zdrojový kód. Toto prostředí se v poslední době stalo hlavním běhovým prostředím pro produkty společnosti Microsoft (například nový operační systém Microsoft Vista je postaven na jeho základě). Tato komponenta je od roku 2002 součástí některých distribucí operačních systémů Microsoft Windows. V případě její absence ji lze zdarma doinstalovat například s využitím služby Windows Update, a to i do starších verzí systému(od Windows 98 výše). Struktura Microsoft .NET Framework je uvedena na obrázku 1.

Obrázek 1:



Common Language Runtime poskytuje virtuální stroj pro vykonávání programů, umožňující programovat bez konkrétních znalostí o CPU na kterém výsledný program poběží. Dalšími funkcemi jsou například správa paměti, obhospodařování vyjímek a programová bezpečnost. Architektura *Common Language Runtime* obsahuje následující hlavní prvky:

- *Common Type System* je základním prvkem pro spolupráci mezi podporovanými jazyky. Definuje ve formě objektů základní datové typy, které jsou použitelné ve všech podporovaných jazycích (každý z nich však může definovat navíc své vlastní). Pokud tedy v programu používáme jen tyto základní typy, lze libovolně přecházet od jednoho jazyka k jinému.
- *Common Language Specification* je sada pravidel a vlastností, kterými se musí vyznačovat jazyk pdporovaný v rámci platformy .NET.
- Common Intermediate Language představuje nejnižší člověku srozumitelný jazyk v infrastruktuře .NET Framework, do něhož jsou překládány programy napsané v jakémkliv základním podporovaném jazyce. Tento jazyk je čistě zásobníkově orientovaný a teprve z něj je dalším překladem získám bytekód.

- *Just In Time Compiler* je překladač využívající pro zvýšení výkonu systému techniku dynamického překladu, pži níž jsou části bytekódu programu překládány no strojového jazyka daného procesoru až v rámci běhu programu.
- *Virtual Execution System* definuje prostředí pro vykonávání kódu v prostředí .NET. Hlavním úkolem je poskytovat podporu pro vykonávání instrukcí Common Intermediate Language.

2 Rozbor problematiky

2.1 Vlastnosti systémů inteligentní elektroinstalace

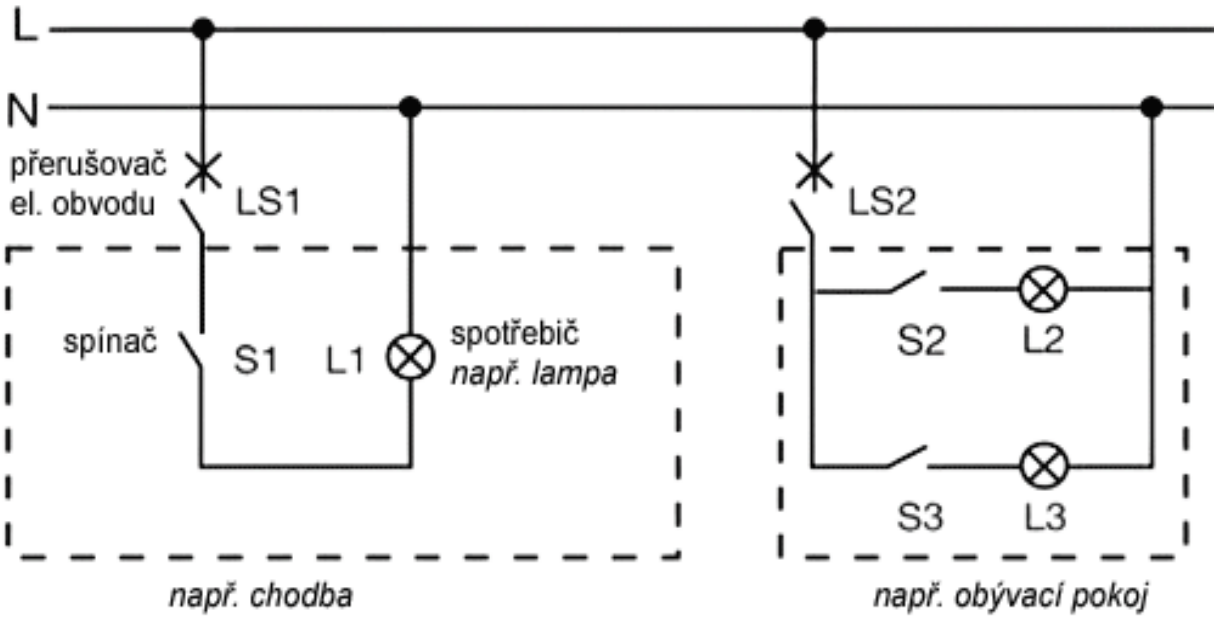
Systémy inteligentních elektroinstalací našly zprvu uplatnění zejména v průmyslových a veřejných budovách, ovšem v posledních letech se díky jejich kvalitám začínají, byť pouze pozvolna, nasazovat také při stavbě obytných domů. Všeobecný rostoucí trend využívání automatizace v domácnostech a v kancelářích se však jen pozvolna přenáší také do návrhu elektroinstalací tohoto typu staveb.

Tento fakt je poněkud překvapivý, uvážíme-li, že činnosti spojené s ovládáním domácí elektroinstalace, typicky osvětlení, jsou velmi četné (určitě například častěji saháme na vypínač osvětlení než na ovladač videorekordéru). Od elektroinstalace vlastně neočekáváme, že by se přizpůsobovala našim požadavkům tak jako třeba ovládání televizoru. Jsme zvyklí večer než usneme vstát z postele a zhasnout světla, před odchodem z domu se ode dveří ještě vrátíme zkontrolovat, zdali jsme zhasli všechna svítidla a podobně. Tyto rutinní činnosti lze ale téměř úplně přesunout na systém funkčních jednotek, spravujících signály od čidel a vypínačů rozmístěných v budově, který podle předem definovaného klíče vykoná všechny potřebné akce za nás. Velmi důležitým pozitiven je skutečnost, že z hlediska každodenního užívání není ovládání takového systému v podstatě nijak odlišné od ovládání klasického, jen s tím rozdílem, že poskytuje v souvislosti s možností kombinování a řetězení funkcí výrazně vyšší uživatelský komfort a možnosti snadnějšího přizpůsobení či změn funkcí, ve srovnání s klasickým elektrickým rozvodem.

2.2 Principy funkce klasické elektroinstalace:

V současnosti běžně, a můžeme říci téměř výhradně užívaným, typem elektrických instalací je klasická elektroinstalace, jak ji známe již po mnoho desítek let. Struktura je tvořena pouze silovým vedením, přičemž ovládání jednotlivých spotřebičů je prováděno prostým spínáním a rozpínáním okruhu jejich napájení, jak je uvedeno na obrázku 2.

Obrázek 2:



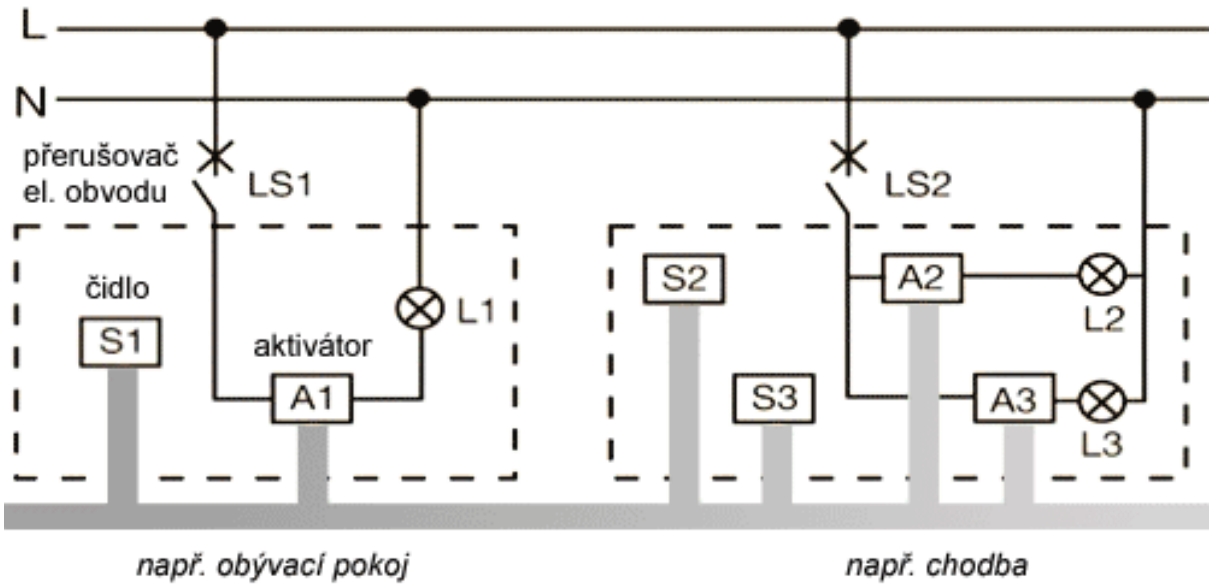
Tato struktura je schopna plnit pouze základní funkce, tedy sepnutí respektive vypnutí přívodu elektrické energie. Navíc tento princip nedovoluje změnit, a to ani částečné, funkce systému bez nutnosti změn zapojení nehledě na to, že realizace některých běžně užívaných funkcí je přinejmenším složitá (například pro ovládání jednoho spotřebiče z více míst je nutné vést do každého takového místa minimálně dva silové vodiče, případně použít speciálních typů přepínačů, čímž se celý systém kabeláže neúměrně komplikuje a tedy i prodražuje).

2.3 Principy funkce inteligentní elektroinstalace:

Systémy tzv. inteligentní elektroinstalace, převážně pracující na principu sběrnicových systémů, přináší do oblasti domácích elektroinstalací nový pohled. Jejich společným znakem je rozdělení funkce klasického vypínače na řídicí a ovládací část – senzor (pro uživatele je reprezentován kolébkou vypínače, ale v principu může jít o jakýkoliv typ čidla), sloužící k detekci požadavků na systém a

jejich vyhodnocení, a výkonovou část – aktor , která zajišťuje samotné sepnutí obvodu spotřebiče. Tato struktura je uvedena na obrázku 3.

Obrázek 3:



Velkou výhodou při projektování těchto systémů je skutečnost, že není zapotřebí přesně definovat, která svítidla či jiné spotřebiče mají být z daného místa ovládány. Důležitý je pouze počet silových obvodů v jednotlivých místnostech a místa, odkud by tyto obvody měly být ovládány. Konkrétní požadavky ovládání, tj. jaký senzor je určen pro ovládání příslušných spotřebičů s požadovanou funkcí, se do systému vkládají prostřednictvím konfigurace funkcí jednotlivých funkčních jednotek, případně centrální řídící jednotky. Způsob kladení kabelových rozvodů pro sběrnicové systémy je zcela odlišný od postupu klasické elektroinstalace. K senzorům (spínačům) je třeba uložit vedení datové sběrnice, které jsou velmi často vedeny společně se silovým vedením. Díky možnosti umístit aktivní prvky přímo ke spotřebičům tak lze výrazně zjednodušit systém silového vedení (postačí teoreticky jeden okruh ke kterému jsou spotřebiče podle potřeby připojovány či odpojovány). Tento způsob instalace je tedy kvůli odlišnému způsobu vedení kabeláže vhodný pouze pro novostavby nebo při celkových rekonstrukcích budov.

Mezi výhody takového uspořádání patří zejména možnost pružně upravovat funkce systému podle aktuálních potřeb prostým přeprogramováním funkcí systému, spínání spotřebičů nezávislé na zapojení a s tím související spínaní z libovolného počtu vstupů nebo jejich kombinací, možnost naprogramování funkcí, které pak systém vykonává automaticky, možnost použití logických a časových funkcí a

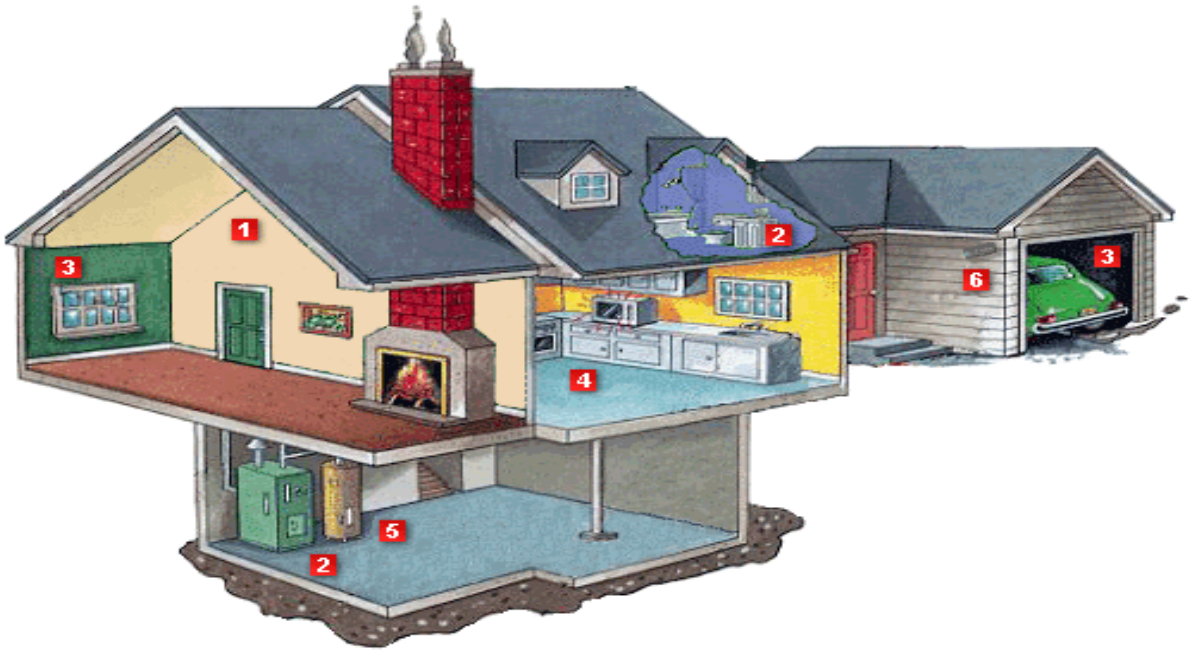
v neposlední řadě také výrazné usnadnění monitoringu a případného zobrazování aktuálních stavů jednotlivých prvků.

2.4 Možnosti systémů inteligentní elektroinstalace

Systém inteligentní elektroinstalace umožňuje téměř libovolně definovat funkce jednotlivých ovládacích prvků skrze změny v konfiguraci systému. Konfiguraci jednotek, stejně jako monitoring jejich okamžitého stavu, lze provádět jak místně, tak dálkově prostřednictvím počítače či mobilního telefonu. Díky zpracování signálů z čidel (vypínačů osvětlení, čidel přítomnosti osob a podobně) jako vstupních signálů pro logickou řídící jednotku, lze snadno vyhodnocovat také další parametry těchto signálů než jen prostou změnu jejich stavu. Tak lze například přiřadit jednomu ovládacímu prvku více rozličných funkcí. Která z funkcí se má skutečně vykonat se pak vyhodnocuje například z doby sepnutí ovládacího prvku či počtu sepnutí v rámci daného časového intervalu. Funkce ovládacího prvku může být také závislá na splnění rozličných podmínek definovaných uživatelem.

Tyto vlastnosti umožňují vytváření komplikovaných a provázaných funkcí zvyšujících pohodlí uživatele. Příkladem mohou být tzv. *centrální funkce.* Typickým použitím je vypnutí všech svítidel a určených spotřebičů při odchodu z domu. Svou podstatou umožňuje tento princip ovládání zahrnout také další podsystémy v rámci objektu a vytvořit tak integrovaný systém kompletně pokrývající ovládání v celém domě,jak je naznačeno na obrázku 4.

Obrázek 4:



1. Ovládání a regulace osvětlení - automatická regulace osvětlení podle přítomnosti osob, denní doby, simulace přítomnosti, centrální vypínání a zapínání spotřebičů atd.
2. Řízení kotelen, regulace vytápění a klimatizace po jednotlivých místnostech, různé časové programy
3. Ovládání žaluzií, markýz, rolet a garážových vrat - ruční nebo automatické podle denní doby
4. Správa spotřebičů a celého systému ovládacími panely, případně telefonem či dálkovým ovladačem
5. Regulace tepelných čerpadel, kontrola a řízení spotřeby energie, dálkový odečet elektroměrů, měřičů tepla
6. Kamerové a zabezpečovací systémy, propojení s dalšími systémy jako UPS, připojení kontrolních prvků úniku plynu nebo vody, elektronické domovní zámky

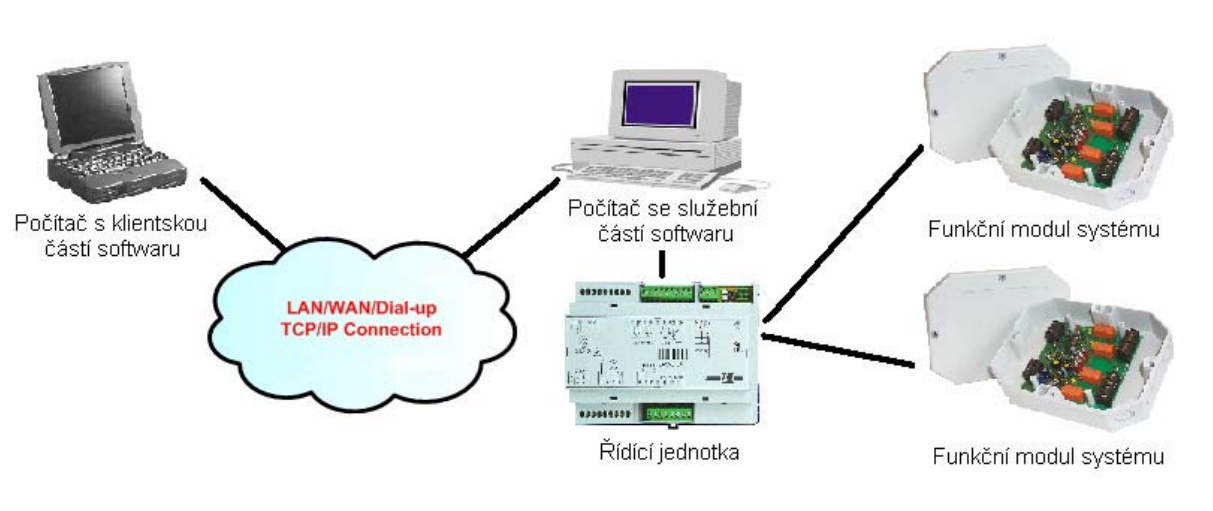
V neposlední řadě je nutné si uvědomit, že použití těchto inteligentních systémů, ačkoliv jejich pořizovací náklady jsou v porovnání s klasickou konfigurací výrazně vyšší, přináší bezpochyby vedle zvýšení komfortu také nezanedbatelnou úsporu spotřeby energií. Ani při nejvyšším úsilí není v lidských silách regulovat při zachování požadovaného obytného komfortu spotřebu energie v domě tak, jak to obstará soustava inteligentních elektroinstalačních komponentů vykonávajících definované funkce.

3 Konkrétní řešení zadaného problému

3.1 Struktura navrženého konfiguračního softwaru

Cílem práce bylo vytvořit software, který umožní konfiguraci systému inteligentní elektroinstalace pracujícího s různými typy funkčních jednotek a to jak místně, z počítače přímo připojeného k vstupnímu bodu systému, tak přes internet. Struktura elektroinstalačního systému, pro kterou byl konfigurační systém navrhován (ovšem není na ní závislý a může pracovat i se strukturami jinými) byla zadána vedoucím práce a je uvedena na obrázku 5.

Obrázek 5:



Jak je z obrázku patrné, je vlastní systém tvořen centrální řídící jednotkou, k níž jsou připojeny jednotlivé funkční moduly, reprezentující vypínače a další čidla potřebná k ovládání osvětlení, případně dalších napojených podsystémů. Řídící jednotka je spojena s počítačem, který slouží jako vstupní bod pro komunikaci s řídící jednotkou. Klienti pak k tomuto počítači přistupují jako k zprostředkovatelskému serveru prostřednictvím standardního spojení využívajícího protokol TCP/IP.

Hlavním požadavkem při návrhu bylo zachovat maximální možnou míru obecnosti v přístupu k funkčním modulům systému a jejich vlastnostem tak, aby použití konfiguračního softwaru nebylo prakticky závislé na konkrétní struktuře a hardwarovém vybavení konfigurovaného systému. Z tohoto důvodu byl při tvorbě softwaru použit koncept definičních souborů pro popis vlastností jednotek. Pro zachování obecnosti při konfiguraci pak byl zvolen postup, při kterém uživatel postupně definuje základní prvky a jejich vzájemné vztahy, které dohromady určují funkci systému jako celku. Těmito základními prvky, jejichž struktura je již pevně dána, jsou :

- Podmínky

  Podmínka vyjadřuje požadavek na specifický setrvalý stav určitého parametru (například světlo v obývacím pokoji je zapnuté). Její výsledek je vždy buď splněno, nebo nesplněno. Aby bylo možné definovat i složité funkce, je systém vybaven i pro práci s podmínkami které jsou vázány na stav vyššího počtu parametrů jednoho či různých modulů.

- Akce

  Akce vyjadřují jednorázový požadavek na změnu stavu konkrétního parametru (například zapni světlo v garáži). Systém je vybaven také pro vytváření zřetězených akcí, které mění stav celé řady parametrů, což je důležité hlavně při konfiguraci tzv. centrálních funkcí.

- Události

  Událost vyjadřuje změnu stavu určitého parametru a podobně jako podmínka nabývá jen dvou hodnot nastala a nenastala. Na rozdíl od podmínky však nejde o stavovou, nýbrž hranovou záležitost („pokud světlo svítí“ je dotaz na stav a tedy podmínka, ale „pokud se světlo právě rozsvítilo“ je dotaz na změnu stavu a tedy událost). Je tedy nutné si uvědomit, že událost nastává jen v okamžiku požadované změny a nemůže tedy mít trvalý pozitivní stav.

  Pro potřeby systému byl tento prvek rozšířen a je využíván jako spojovací prvek mezi podmínkami a akcemi. Událost jak je popsána v předchozím odstavci je tak doplněna o seznam jí příslušejících podmínek a akcí. V okamžiku události jsou pak vyhodnoceny podmínky dle seznamu podmínek a v případě jejich splnění je vykonána sekvence ze seznamu akcí.

Jelikož nelze v programu uchovávat detailní informace o konkrétních vlastnostech všech použitelných jednotek, nehledě na to, že takový přístup by vyžadoval změnu programu vždy po uvedení nového typu jednotky do provozu, bylo nutné použít koncepci, ve které jsou vlastnosti jednotek definovány mimo vlastní program, kde jsou v případě potřeby k dispozici.

Z celé řady možností byl zvolen princip založený na definičních souborech jednotlivých modulů. Informace o vlastnostech každého modulu jsou definovány v souboru typu XML, který bude dodáván výrobcem společně s modulem. Tento soubor, jehož formát je popsán v samostatné kapitole, obsahuje detailní informace o struktuře modulu a jeho funkčnosti a poskytuje tak všechny informace potřebné ke konfiguraci jeho činnosti v rámci celého systému.

Vlastní konfigurační software, k jehož vytvoření byl použit jazyk C# v rámci vývojové platformy Microsoft .NET, je z funkčního hlediska rozdělen na dvě základní části:

- část klientskou, tedy aplikaci sloužící uživateli ke konfiguraci požadované funkcionality systému v uživatelsky přívětivém grafickém prostředí. Tato část může běžet nezávisle na kterémkoli počítači. Součástí jsou i funkce pro vytvoření souboru reprezentujícího požadovanou konfiguraci ve formátu srozumitelném pro řídící jednotku a jeho odeslání přes Internet na cílový počítač spolupracující s řídící jednotkou.
- část služební, která přijímá informace od autorizovaných klientských aplikací a předává je řídící jednotce systému inteligentní elektroinstalace.

Obě části budou detailně popsány s samostatných kapitolách.

3.2 Struktury použitých souborů pro uchování dat

3.2.1 Struktura definičního XML souboru modulu

Tento soubor slouží jako zdroj informací o konkrétním typu jednotky, se kterou má systém pracovat. Obsahuje informaci o hardwarové struktuře modulu (počty vstupů a jejich adresy, vnitřní proměnné a podobně), jeho funkčnosti (jakých činností je schopen), ale také o způsobu zobrazení jednotlivých parametrů při práci s modulem v konfiguračním softwaru. Jak je pro dokumenty jazyka XML obvyklé, jedná se o hierarchickou strukturu značek, nazýváme je tagy, které reprezentují konkrétní vlastnosti příslušného modulu.

Základními stavebními kameny definičního souboru jsou parametry, reprezentované tagem *param.* Definice každého parametru obsahuje ve formě atributů jazyka XML následující informace:

- Typ objektu – je reprezentován hodnotou atributu *type*; tento atribut je povinný a určuje jaký typ dat parametr obsahuje
- Práva uživatele v nakládání s hodnotou parametru – reprezentována hodnotou atributu *access*. V tomto případě přichází v úvaho pouze dvě hodnoty: R/W -povoleno čtení i zápis, R- povoleno pouze čtení
- Identifikační řetězec – je reprezentován hodnotou atributu *id*. Určuje zda je parametr potenciálním zdrojem podmínky či události, nebo cílem akce.

Další informace o parametru jsou obsaženy s potomcích tagu *param*. Složení těchto prvků je pro daný typ parametru pevně dáno. Zvláštní postavení mají dva prvky, které se mohou, ovšem vždy společně, vyskytnout u kteréhokoliv typu parametru. Jsou to *oid* (Object ID) který specifikuje hardwarovou adresu parametru v rámci modulu a *osize*, který specifikuje paměťový prostor parametru přidělený. Parametry, které ve své definici tuto dvojici tagů neobsahují, slouží pouze k usnadnění orientace uživatele ve struktuře modulu (jde například o přidělování vlastních názvů jednotlivým vstupům, což nemá na jejich funkci žádný vliv).

Přehled používaných typů parametrů, včetně popisu a příkladů jejich definic :

- Typ byte/word

  Jedná se o parametry, které nabývají číselných hodnot. Liší se pouze ve velikosti přidělené paměti v modulu, přičemž typu byte je přidělen prostor o velikosti1 byte, typu word pak 2 byty.

  Definice:

```
<param type=”word” access=”R/W”>
    <min>1<min>                    // minimální hodnota parametru
    <max>1000<max>                 //maximální hodnota parametru
    <default>500<default>          // přednastavená hodnota parametru
    <name>Doba sepnutí<name> //název parametru
    <style>1<style>                //styl zobrazení parametru
    <oid>13<oid>                   //adresa parametru v rámci modulu
    <osize>2<osize>                // velikost paměťi přidělené parametru v modulu
</param>
```

- Typ enum

  Jedná se o parametry, které nabývají jedné z předdefinovaných hodnot definovaných hodnotami tagů *item*.

  Definice:

```
<param type=”enum” access=”R/W”>
    <item val=”0”>Povoleny</item>    //předdefinovaná hodnota parametru
    <item val=”1”>Zakázány</item>   // předdefinovaná hodnota parametru
    <default>1<default>              // přednastavená hodnota parametru
    <name>Povolení změn<name>   //název parametru
    <style>1<style>                 //styl zobrazení parametru
    <oid>12<oid>                    //adresa parametru v rámci modulu
    <osize>5<osize>                // velikost paměťi přidělené parametru v modulu
</param>
```

- Typ string

  Jedná se o parametry, jejichž hodnoty mají tvar znakových řetězců.

  Definice:

```
<param type=”string” access=”R/W” id=“C1“>
    <maxlength>20</maxlength>              //maximální délka řetězce
    <default>Čidlo pohybu<default>         // přednastavená hodnota parametru
    <name>Čidlo pohybu<name>               //název parametru
    <style>1<style>                        //styl zobrazení parametru
</param>
```

- Typ separator

  Jedná se o parametr který nemá žádný význam pro vlastní konfiguraci ani spojitost s vlastnostmi modulu. Jeho jediným účelem je zpřehlednit zobrazení seznamů parametrů v grafickém prostředí (je interpretován jako prázdná pozice s barvou definovanou jeho vnitřním tagem *style*) .

  Definice:

```
<param type=”separetor” >
    <style>1<style>                  //styl zobrazení parametru
</param>
```

- Typy name, adress

  Jedná se o parametry, jejichž hodnoty jsou shodné s již uvedenými typy (s typem string v případě name a s typem word v případě adress), ovšem jejich úloha v dokumentu je specifická. Tyto parametry jednoznačně určují modul v rámci celého systému a proto je třeba aby každý modul obsahoval každý z těchto atributů právě jednou a aby hodnoty parametrů těchto typů byly v rámci celého systému unikátní.

Základní struktura definičního souboru je následující:

```
<unit>
   <unitname>Vypinac typ V156</unitname>
   <general>….</general>
   <conditions>….</conditions>
   <actions>…..</actions>
   <events>….</events>
</unit>
```

Tag *unit* vytváří mateřský uzel XML dokumentu a sdružuje pod sebou pět přímých potomků.

První z nich má název *unitname*. Tento tag nemá žádné další potomky a jeho hodnota definuje typové označení, pod kterým bude jednotka vystupovat vůči uživateli. Každý z následující čtveřice tagů definuje vlastnosti jednotky ve specifické oblasti a to následovně:

- *general*- definuje vstupy, výstupy a další vnitřní proměnné modulu.
  Vnitřní struktura je následující:

```
<general>
    <tab>                                          //skupina příbuzných parametrů
      <caption>Hlavní parametry</caption>          //název skupiny
      <param type=”name” access=”R/W”>…. </param>  //parametr
       <param type=”adress” access=”R/W”>….</param> //parametr
        ….
    </tab>
    ….
  </general>
```

Tag *general* je tvořen libovolným počtem tagů *tab*, reprezentujících skupiny příbuzných parametrů. Každý z těchto tagů obsahuje potomka *caption*, jehož hodnota specifikuje název skupiny, a libovolný počet parametrů. Toto dělení parametrů do skupin nemá přímý vliv na funkce systému elektroinstalace, ale slouží k přehlednějšímu zobrazení parametrů modulu při práci v navrženém grafickém prostředí.

- *conditions*- definuje podmínky, které mohou být na základě stavu tohoto konkrétního modulu, přesněji řečeno jeho parametrů, vyhodnocovány (například vstup číslo 1 je na logické jedničce)

  Vnitřní struktura je následující:

```
<conditions>
    <conditionsource>                    //zdroj podmínky bez reference na parametr
        <val>Napájení</val>                    //název zdroje podmínky
        <cond>                                 //konkrétní podmínka
            <type>Vstup neaktivní</type> //název podmínky
            <oper>ZE</oper>                    //operace  nutná k vyhodnocení
        </cond>
        <cond>….</cond>
        ….
        <oid>3</oid>                   //adresa zdroje podmínky v rámci modulu
        <osize>2</osize>               //velikost  zdroje podmínky v bytech
    </conditionsource>
    <conditionsource ref="I1">         //zdroj podmínky  s reference na parametr
        <cond>                         //konkrétní podmínka
            <type>Sepnuto</type>       //název podmínky
            <oper>NZ</oper>            //operace  nutná k vyhodnocení
            <param type="word">…</param> //parametr podmínky
            ….
        </cond>
        ….
        <oid>13</oid>        //adresa parametru který je zdrojem podmínky
        <osize>2</osize>     //velikost  parametru který je zdrojem podmínky
    </conditionsource>
    …..
</conditions>
```

Tag *conditions* je tvořen libovolným počtem tagů *conditionsource*, reprezentujících potenciální zdroje podmínek, které lze pro tento modul vyhodnocovat. Takovým zdrojem může být jeden z parametrů modulu definovaných v sekci *general* ale může jít také o zdroj jiného typu (například stav napájení modulu). Tyto dvě možnosti jsou, jak je patrné z příkladu výše, v definičním souboru rozlišeny pomocí atributu *ref* tagu *conditionsource.* Přítomnost tohoto atributu značí, že zdrojem informace pro vyhodnocení

podmínky je jeden z definovaných parametrů, její nepřítomnost pak že zdroj je jiného typu, přičemž ten je pak určen hodnotou tagu *val* .

Vnitřní struktura tagu *conditionsource* sestává z libovolného počtu tagů *cond,* reprezentujících konkrétní podmínky, z nichž každý obsahuje název podmínky(hodnota tagu *type*), operaci jejíž aplikací na data zdroje je podmínka vyhodnocena (hodnota tagu *oper*; možnými hodnotami jsou ZE-hodnota je nulová, NZ-hodnota není nulová, GR-hodnota je větší než, LE-hodnota je menší než) a případně libovolné množství dalších parametrů, sloužících pro bližší určení podmínky (například definují požadovanou hodnotu pro operace porovnání při vyhodnocování podmínky). Tyto parametry jsou definovány způsobem uvedeným na začátku této kapitoly, přičemž však nemohou obsahovat atributy *access* (implicitně se u nich předpokládá hodnota R/W) ani *id.*

Posledními dvěma potomky každého tagu *conditionsource* jsou tagy *oid* (definuje umístění zdrojových dat v modulu) a *osize* (definuje velikost zdrojových dat v bytech)

- *actions*- definuje akce, které mohou být modulem nad jeho parametry vykonány (například sepni výstup číslo 1)

  Vnitřní struktura je následující:

```
<actions>
    <target ref="I1">                          //cíl akce  s referencí na parametr
        <ack>                                  //konkrétní podmínka
            <type>Sepni</type>                 //název podmínky
            <oper>NZ</oper>                    //operace  k provedení
            <param type="word">…</param> //parametr akce
            ….
        </ack>
        ….
        <oid>18</oid>                          //adresa parametru který je cílem akce
        <osize>2</osize>                       //velikost  parametru který je cílem akce
    </target>
    …..
</actions>
```

Struktura je analogická se strukturou tagu *conditions,* s tím rozdílem, že jelikož cílem akce mohou být pouze parametry modulu, musí být tag *target* vždy definován s atributem *ref*.

- *events*- definuje události, které mohou nad parametry dané jednotky nastat (například právě došlo ke změně stavu čidla 1 z logické nuly na logickou jedničku)

  Vnitřní struktura tohoto tagu je naprosto analogická s tagem *conditions.* Z tohoto důvodu není samostatný příklad uveden.

3.2.2 Struktura rozšířeného XML konfiguračního souboru

Tento dokument slouží pro ukládání dat o konfiguraci systému a způsobu jejího zobrazení v grafickém prostředí. Jeho obsah je měněn průběžně se změnami prováděnými uživatelem v grafickém prostředí a jeho obsah tak vždy přesně odráží aktuální funkčnost systému.

Celý dokument je rozdělen na čtyři základní oddíly a to následovně:

```
<settings>
        <units>…</units>                  //oddíl definující moduly systému
        <conditions>….</conditions>   //oddíl definující podmínky
        <actions>….</actions>           //oddíl definující akce
        <events>….</events>            //oddíl definující události
</settings>
```

Význam jednotlivých oddílů a jejich vnitřní struktury je uveden v následujícím přehledu:

- Tag *units* obsahuje informace o modulech, které uživatel v konfiguraci používá. Vnitřní struktura je tvořena seznamem tagů *unit,* které jsou obrazy definičních souborů konkrétních typů modulů. V těchto obrazech je každému parametru v sekci *general* přidán tag *value,* jehož vnitřní hodnota určuje aktuální hodnotu tohoto parametru v aktuální konfiguraci.
- Tag *conditions* uchovává informace o podmínkách definovaných v rámci systému. Vnitřní struktura je tvořena seznamem tagů *condition* reprezentujících uživatelem nadefinované podmínky. Vnitřní struktura těchto tagů je následující:

```
<condition>
        <general>
                <name>Název podmínky</name>   //název podmínky
                <overview>funguje</overview>       //popis
                <enabled>Y</enabled>               //způsob vyhodnocení podmínky
                <logoper>A</logoper>               //llog. operace pro podpodmínky
        </general>
        <subconditions>
                <subcondition>                                    //konkrétní podpodmínka
                        <sourceunit>Modul 3</sourceunit >         //zdrojová jednotka
                        <sourceparam>Vstup 1</sourceparam > //zdrojový parametr
                        <type>Vypnuto</type >                     //jméno podpodmínky
                        <oper>ZE</oper >                          //operace k vyhodnocení
                        <param type="word">….</param>
                        ….
                <subcondition>
                ….
        <subconditions>
</condition>
```

Jak je z příkladu výše patrné, je každá podmínka určena čtveřicí hodnot definovaných v sekci *general* a libovolným počtem vložených podpodmínek, reprezentovaných tagy *subcondition*, uvedených v sekci *subconditions.*

Hodnoty v sekci *general* určují vlastnosti podmínky jako celku. Najdeme zde název podmínky (tag *name*), popis podmínky (tag *ovwrview*), určení jak přistupovat k vyhodnocení podmínky (tag enabled, hodnota Y značí vyhodnocovat podle stavu podpodmínek, hodnota 1 vyhodnotit vždy jako splněno a hodnota 0 vyhodnotit vždy jako nesplněno) a určení jaký stav podpodmínek vede ke splnění podmínky (tag *logoper,* hodnota A značí, že všechny podmínky musí být splněny, hodnota O pak že postačuje splnění jediné).

Každá podpodmínka je definována svým jménem (tag *type*), parametrem jehož stav je předmětem dotazu (tag *sourceparam*), modulem kterému tento parametr náleží (tag *sourceunit*) a operací určující jaký dotaz má být nad parametrem vyhodnocen (tag oper, hodnota ZE značí parametr je nulový, hodnota NZ značí parametr je nenulový, hodnota GR značí parametr je větší než, hodnota LE značí parametr je menší než), V případě že v sobě podpodmínka definuje další parametry, jsou tyto uvedeny standardním

způsobem , popsaným v kapitole o definičním souboru modulu, za tagem *oper,* přičemž hodnota atributu access je implicitně uvažována R/W a není uváděna.

Již struktura napovídá že výsledná hodnota podmínky, která může být vždy jen splněno či nesplněno, může být díky systému podpodmínek, pojatých jako specifický dotaz na stav konkrétního parametru jednoho z modulů, závislá na neomezeném počtu parametrů libovolných modulů systému. Což umožňuje snadné vytváření komplikovaných funkcí.

- Tag *actions* uchovává informace o akcích definovaných v rámci systému. Vnitřní struktura je analogická s tagem *conditions,* s drobnými odlišnostmi. Hodnoty tagu *enabled* jsou Y pro povolení provedení akce a N pro jeho zakázání. Jelikož podakce, které jsou opět chápány jako příkazy určující změnu konkrétního parametru určené jednotky, jsou vykonávány vždy všechny jako sekvence příkazů, není zde potřebný tag logoper.

  Akce tedy definuje posloupnost změn jednotlivých parametrů jež se má provést při jejím spuštění.

- Tag events uchovává informace o událostech definovaných v rámci systému. Vnitřní struktura je tvořena seznamem tagů *event* reprezentujících uživatelem nadefinované události. Vnitřní struktura těchto tagů je následující:

```
<event>
    <general>
        <name>Název události</name>                    //název události
        <overview>nefunguje</overview>                 //popis
        <logoper>O</logoper>                           //operace mezi podmínkami
        <eventsourceunit>Modul 6</eventsourceunit> //zdrojový modul události
        <eventsource>Vstup 2</eventsource>             //zdrojový parametr události
        <eventtype>Sepnul</eventtype>                  //událost
        <oper>NZ</oper>                                //operacek vyhodnocení události
        <param type="word">…</param>                   //parametr události
        ….
    </general>
```

```
        <subconditions>
                <subcondition>
                        <name>Venku je tma</name>    //název vyhodnocované podmínky
                </subcondition>
                ….
        </subconditions>
        <subactions>
                <subaction>
                        <name>Zapni světla</name>     //název akce k vykonání
                </subaction>
                ….
        </subactions>
</event>
```

Z příkladu je patrné že popis každé události sestává ze tří základních částí.

První část, reprezentovaná tagem *general*, popisuje vlastnosti události jako celku. Obsahuje informaci o názvu události (tag *name*), popisu události (tag *overview*), stavu podmínek vedoucímu k vykonání akcí (tag *logoper,* hodnota A značí,že všechny podmínky musí být splněny, hodnota O pak že postačuje splnění jediné). Dále definuje změnu stavu konkrétního parametru která je podnětem k provedení události. Tato změna je definována svým typem (tag oper), parametrem na kterém nastává (tag *eventsource*) a jednotkou k níž tento parametr náleží (tag *eventsourceunit*).

Druhá část, reprezentovaná tagem *subconditions* definuje seznam podmínek které se mají vyhodnocovat pokud nastane změna parametru definovaná v tagu *general.*

Třetí část reprezentovaná tagem *subactions* definuje seznam akcí, které se mají vykonat, pokud je výsledek vyhodnocení podmínek uvedených v tagu *subconditions* kladný.

Z výše uvedeného je patrné, že události jsou prvkem, který spojuje uživatelem definované podmínky a akce ve funkční celky s uživatelem definovaným spouštěcím signálem. Díky možnosti kombinovat předem definované podmínky, které jsou již samy o sobě kombinacemi elementárních podmínek, a navázat na ně libovolnou sekvenci předem definovaných akcí je možné takto vytvářet velmi komplikované funkce.

3.2.3 Struktura finálního textového konfiguračního souboru

Tento soubor reprezentuje souhrn informací pro řídící jednotku oproštěný od přebytečné informace, která se netýká přímo instrukcí pro její činnost. Pro tento soubor byl pro zachování alespoň základní přehlednosti zvolen textový formát, kdy dokument je tvořen seznamem funkčních bloků se specifickým významem a pevnou strukturou. Každý z bloků reprezentuje jeden ze základních prvků konfigurace, přičemž typ tohoto prvku je definován písmenem stojícím v prvním řádku bloku. Koncem bloku je další řádek obsahující jedno písmeno, definující zároveň typ následujícího bloku. Bloky jsou řazeny podle typu v pořadí moduly, podmínky, akce, události. Pořadí prvků stejného typu je pak určeno pořadím odpovídajících prvků v rozšířeném konfiguračním souboru (to odpovídá pořadí v jakém byly uživatelem definovány).

Struktura jednotlivých typů bloků je následující:

- Typ unit-popisuje relevantní vlastnosti funkčního modulu

```
U                          //začátek bloku popisujícího modul
100;10;Jednotka venku      //informace o názvu modulu
10;2;100                   //určení adresy  modulu v rámci systému
22;2;500                   //určení  hodnoty dalšího parametru
23;2;365                   //určení  hodnoty dalšího parametru
…..
```

Každý řádek bloku (mimo řádku určujícího typ) definuje hodnotu jednoho z parametrů modulu a to následujícím způsobem:

oid (místo v paměti modulu) parametru; maximální velikost v bytech; hodnota

Přitom první řádek slouží vždy k definici názvu modulu a druhý k určení jeho adresy z pohledu řídící jednotky.

- Typ condition- popisuje vlastnosti konkrétní definované podmínky

```
C                  //začátek bloku popisujícího podmínku
Y;A                //obecné vlastnosti podmínky
5;1;ZE             //podpodmínka  bez parametru
15;55;GR;500       //podpodmínka  s parametrem
....
```

První řádek po určení typu představuje definici obecných vlastností podmínky ve tvaru:

vyhodnocení podmínky; logická operace aplikovaná na množinu podpodmínek

Vyhodnocením podmínky rozumějme následující. Hodnota Y značí běžné vyhodnocení v závislosti na splnění podpodmínek, hodnota 1 značí vždy vyhodnotit podmínku jako splněnou, hodnota 0 značí vždy vyhodnotit podmínku jako nesplněnou. Operace aplikovaná na množinu podpodmínek určuje zda je nutné aby byly splněny všechny podpodmínky naráz (hodnota A), nebo postačí splnění byť jediné podpodmínky (hodnota O).

Následující řádky pak definují podpodmínky (dotazy na stav konkrétních parametrů konkrétních modulů) z nichž je podmínka složena, a to následovně:

adresa modulu; oid zdroje podmínky v rámci modulu ; operátor pro vyhodnocení; parametr;…

Pokud jsou součástí podpodmínky i její parametry, jsou uvedeny za operátorem pro vyhodnocení podpodmínky oddělené středníkem, jak je naznačeno v příkladu.

- Typ action- popisuje vlastnosti konkrétní definované akce

```
A               //začátek bloku popisujícího akci
YE              //povolení/zakázání provedení akce
5;1;ZE          //akce k provedení bez parametru
15;55;SE;500    //akce k provedení s parametrem
...
```

První řádek bloku (po řádku určujícím typ) definuje zda je provedení akce povoleno (hodnota YE) nebo zakázáno (hodnota NO). Každý další řádek definuje změnu jednoho z parametrů nějakého z modulů systému, která se má v rámci akce provést a to následujícím způsobem:

adresa modulu; oid cíle akce v rámci modulu ; operátor určující změnu; parametr;…

Hodnoty operátoru pro určení změny jsou: NZ-nastav na logickou jedničku, ZE-nastav na logickou nulu, SE- nastav hodnotu parametru na určenou hodnotu. Pokud jsou součástí změny i její parametry, jsou uvedeny za operátorem určujícím její typ oddělené středníkem, jak je naznačeno v příkladu.

- Typ event- popisuje vlastnosti konkrétní definované události

```
E                //začátek bloku popisujícího událost
2;25;O;200       //obecné vlastnosti události
1;2;9;12         //seznam všech podmínek
1;3 ;5;7;9       //seznam všech akcí
```

První řádek po určení typu představuje definici obecných vlastností podmínky ve tvaru:

adresa modulu; oid zdroje události; vyhodnocení podmínek; parametr; …

Parametr vyhodnocení podmínek určuje, jakým způsobem je vyhodnocována výsledná hodnota seznamu podmínek, pokud událost nastane. Hodnota A značí, že pro kladné vyhodnocení musí být splněny všechny dílčí podmínky. Hodnota O značí, že pro kladné vyhodnocení je postačující splnění alespoň jedné dílčí podmínky. Hodnota I značí, že podmínky nemají být vůbec vyhodnocovány a vyhodnocení bude vždy kladné.

Druhý řádek je seznamem všech podmínek přiřazených této události, přičemž každé číslo vyjadřuje pořadí bloku popisujícího konkrétní podmínku mezi bloky typu „C“.

Třetí řádek je seznamem všech akcí přiřazených této události, přičemž každé číslo vyjadřuje pořadí bloku popisujícího konkrétní akci mezi bloky typu „A“.

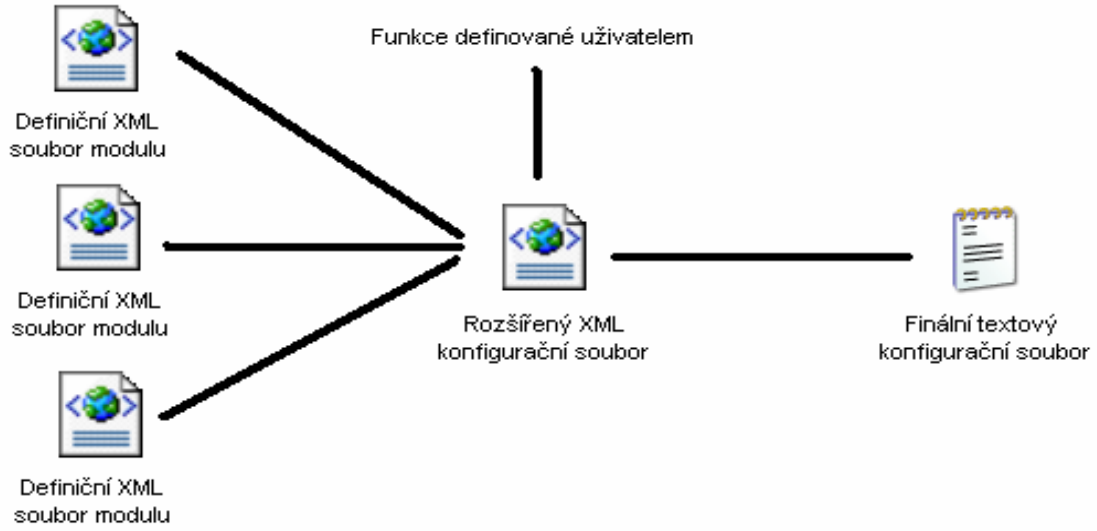
Dokument je po potvrzení správnosti vytvořené konfigurace vytvářen automaticky konverzí z rozšířeného konfiguračního XML dokumentu.

3.3 Klientská část konfiguračního softwaru

3.3.1 Základní struktura klientské části konfiguračního softwaru

Tato část slouží k vytvoření konfiguračního souboru, který detailně popisuje aktuální konfiguraci systému elektroinstalace a je posléze odeslán do jeho řídící jednotky. Jde o aplikaci s uživatelsky přívětivým grafickým rozhraním, které umožňuje uživateli sestavit konfiguraci jím vlastněného systému bez požadavku na hlubší znalost problematiky řízení či programování. Vytvoření finální podoby konfiguračního souboru sestává ze dvou na sebe navazujících kroků, jak je naznačeno na obrázku 6.

Obrázek 6:



V prvním kroku je z definičních souborů, dodaných výrobcem ke každému modulu systému, a požadavků zadaných uživatelem postupně sestavován rozšířený konfigurační soubor, který detailně popisuje celý konfigurovaný systém a to jak po stránce hardwaru (soubor obsahuje detailní popis každého z používaných modulů), tak po stránce funkčnosti (definuje vztahy mezi signály z modulů a na ně navazující funkčnost). Při volbě formátu tohoto souboru byla, stejně jako v případě definičních souborů pro konkrétní moduly, dána přednost formátu XML. Vlastnosti tohoto značkovacího jazyka a jeho výhody jsou popsány s samostatném oddílu práce. Dalším důvodem použití jazyka XML byl fakt, že tento jazyk je srozumitelný a přehledný a při správné volbě struktury umožňuje nahlédnout funkčnost systému i bez speciálního softwarového vybavení. V případě nutnosti lze tedy provádět změny

v konfiguraci systému, či dokonce vytvářet zcela nové konfigurace, v libovolném textovém editoru.

Po dokončení změn v konfiguraci a jejich potvrzení uživatelem je z tohoto rozšířeného formátu, zahrnujícího i informace sloužící výhradně pro uživatelské pohodlí při práci se softwarem, vytvořen zhuštěný textový soubor obsahující pouze informace relevantní pro řízení uvedené ve formátu srozumitelném řídící jednotce, který je následně odeslán přes internet do počítače spolupracujícího s řídící jednotkou.

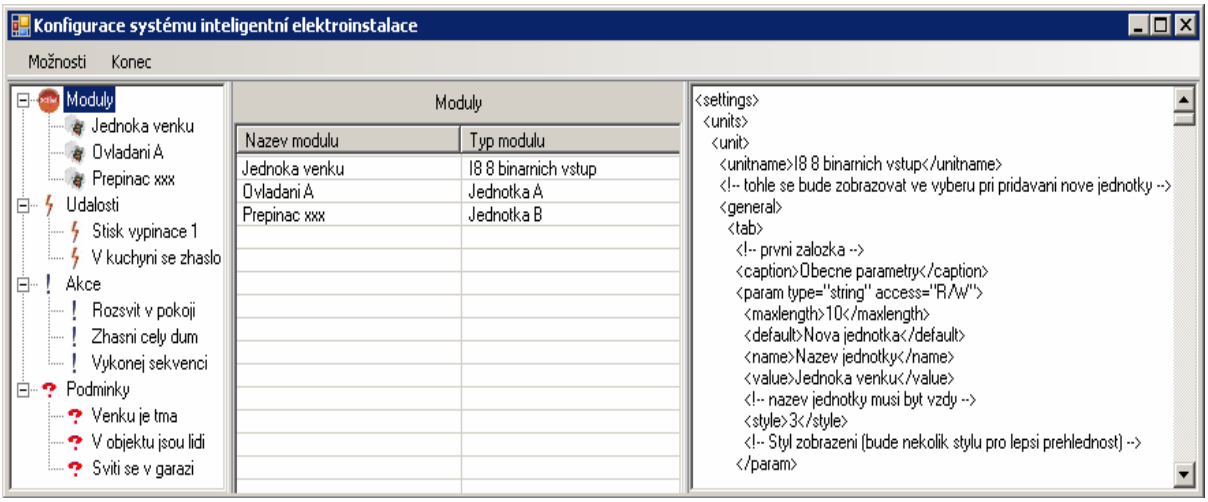
V následujících oddílech je podrobně popsána struktura výše uvedených souborů a také použité principy vytváření konfigurace s použitím navrženého systému.

3.3.2 Grafické rozhraní konfiguračního softwaru a jeho funkce

Grafickcké prostředí je navrženo tak, aby převádělo informace uložené ve formě rozšířeného XML konfiguračního souboru do přehledné grafické formy a umožňovalo provádět změny v konfiguraci systému bez nutnosti znalostí o formátu ukládaných informací, nebo konkrétních vlastností jednotlivých modulů.

Grafické rozhraní je tvořeno třemi částmi (uvedeno na obrázku 7), z nichž dvě jsou základní zatímco třetí je ve výchozím nastavení skryta.

Obrázek 7:



První část, na obrázku zcela vlevo, je tvořena hierarchickým stromem sloužícím k navigaci mezi součástmi systému. Najdeme zde přehledně graficky zaznamenanou strukturu celého systému.

Druhá část, na obrázku uprostřed, je tvořena soustavou tabulek popisujících vlastnosti prvku nebo skupiny prvků aktuálně zvolených v navigačním stromu.

Třetí část, na obrázku zcela vpravo, je ve výchozím nastavení skryta a může být aktivována v hlavním menu aplikace. Tato část zobrazuje aktuální podobu rozšířeného konfiguračního souboru.

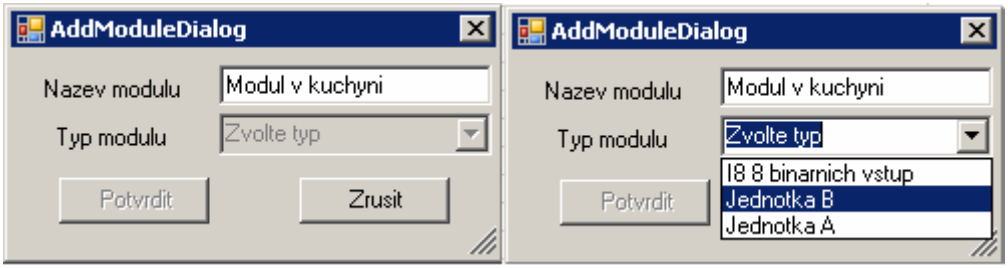
Následuje podrobný popis funkcí popskytovaných jednotlivými částmi aplikace.

- *Stromová struktura* v levé části uživatelské obrazovky neslouží pouze k navigaci mezi prvky systému, ale umožňuje také tyto prvky odebírat a nově definovat pomocí voleb kontextového menu. Kontextové menu je vyvoláno stiskem pravého tlačítka myši na příslušné pozici, od níž se odvíjí nabízená volba. Pokud ke stisku dojde na pozici konkrétního prvku ja nabízeno jeho odebrání, pokud na pozici skupiny prvků (například moduly), pak je nabízeno přidání nového prvku daného typu (v našem případě nového modulu).

  Přidávání jednotlivých prvků je provedeno následovně. Při přidání podmínky, akce nebo události je do konfiguračního souboru okamžitě přidán nový prvek požadovaného typu s názvem ve formátu „Nový prvek“+„aktuální datum a čas“. Tento formát názvu zajišťuje že přidáním prvku nebude narušena jednoznačnost názvů, která je v aplikaci požadována.

  Přidání nového mudulu je zkomplikováno tím, že je nutné přiřadit typ přidávaného modulu. Proto je přidání modulu vyřešeno pomocí samostatného dialogového okna, uvedeného na obrázku 8.

Obrázek 8:



V tomto dialogovém okně uživatel definuje název modulu, u nějž je okamžitě kontrolováno zachování jednoznačnosti (pokud je zadán již existující název, je text zobrazen červeně a není povoleno potvrzení jeho zadání ani potvrzení celého dialogu). Po zadání korektního názvu je umožněna volba typu modulu. Uživatel volí typ z nabídky, která je generována podle dostupných definičních souborů modulů (zobrazovány jsou hodnoty tagu *unitname*) uložených v adresáři jehož umístění volí uživatel v hlavním menu aplikace. Pokud jsou oba tyto parametry správně zadány je povoleno

potvrzení dialogu, po němž následuje okamžité přidání jednotky požadovaného typu a názvu do rozšířeného konfiguračního souboru. Jeho adresa je přitom nastavena na hodnotu o jedna vyšší než je nejvyšší hodnota tohoto parametru u již definovaných modulů (toto opatření zabraňuje porušení jednoznačnosti adres).

Odebírání všech prvků je provedeno následovně. Pokud odebíraný prvek není součástí definice jiného prvku, je přímo odebrán z konfigurace. Pokud je odebíraný prvek součástí definice jiného prvku, je uživatel upozorněn na tento fakt dialogovou volbou s možností zrušení prováděné akce. Po potvrzení je pak odebrán nejen zvolený prvek, ale s ním i všechny jeho výskyty v definicích ostatních prvků.

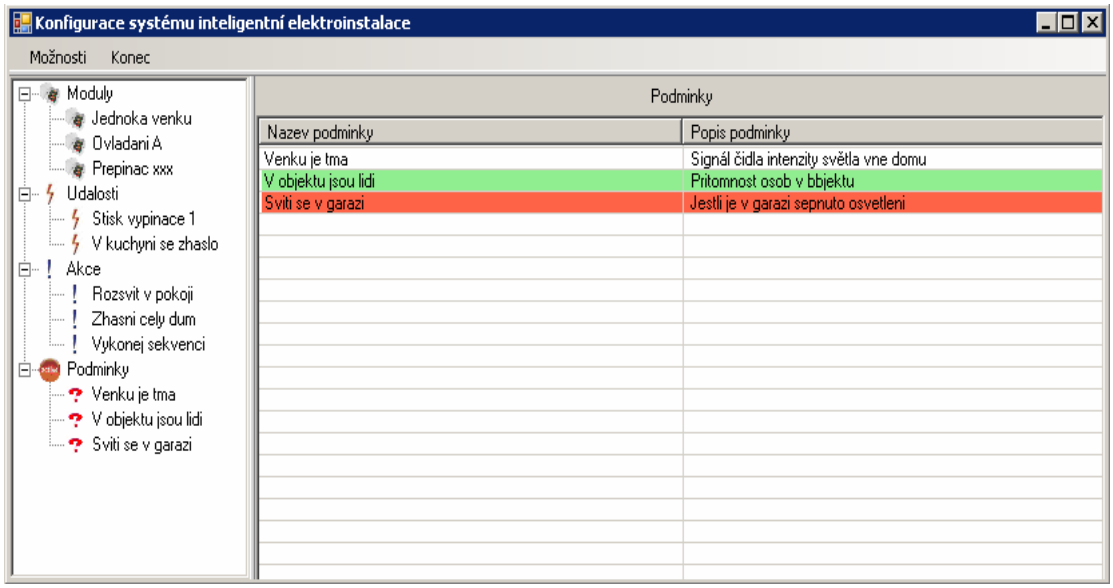
- *Soustava tabulek popisujících vlastnosti aktuálně zvolených prvků* je hlavním nástrojem pro konfiguraci systému, neboť slouží k definici hodnot parametrů a součástí jednotlivých prvků.

  Hodnota parametrů se mění dvojitým kliknutím levým tlačítkem na pole parametru. Pokud je parametr editovatelný je podle jeho typu uvedeného v rozšířeném konfiguračního souboru vyvolán grafický prvek umožňující zadat požadovanou hodnotu (textové pole, výběr čísel, combobox s předdefinovanými hodnotami). Po potvrzení změny je zkontrolováno zda hodnota splňuje definovaná omezení (dodržení délky textového řetězce, či jedinečnost názvů jsou kontrolovány již v průběhu změny). Pokud ano, je změna provedena na příslušném místě konfiguračního souboru, pokud ne je pokus o změnu ignorován (v případě hodnotz číselného parametru mimo povolený rozsah je nastavena bližší z mezních hodnot).

  Obsah i počet tabulek je závislý na tom, který prvek je zvolen v navigačním stromě a proto budou dále popsány všechny možnosti a jejich funkce.

  Nejjednodušší struktura odpovídá situaci kdy je v navigačním stromě zvolena jedna z hlavních skupin (moduly, podmínky, akce, události). Tuto část pak tvoří jediná tabulka se dvěma sloupci. V prvním jsou uvedeny názvy prvků daného typu, při jejichž změně je kontrolováno zachování jednoznačnosti (pokud je zadán již existující název, je text zobrazen červeně a případné potvrzení vede k ignorování změny), v druhém pak jim odpovídající popis (v případě modulů odpovídající typ) jak je uvedeno na obrázku 9.

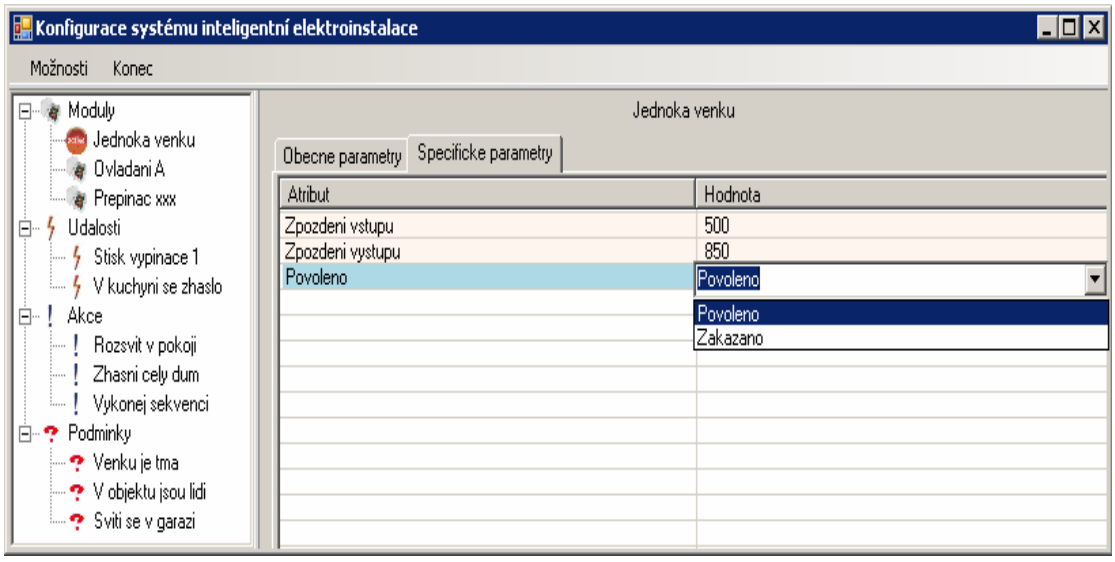
Obrázek 9:



Jak je z obrázku patrné, jsou jednotlivé prvky rozlišeny barvami. Tyto barvy odrážejí hodnotu tagu enabled (u událostí tagu logoper) a slouží ke snadnější orientaci v tom, které prvky jsou povoleny, či zakázány pro vyhodnocení respektive provedení. Zelená barva značí pro podmínku, že tato má být vždy vyhodnocena jako splněná, pro událost pak že podmínky nemají být vyhodnocovány a mají být rovnou provedeny akce. Červená barva značí pro podmínku že tato má být vždy vyhodnocena jako nesplněná, pro akci pak že nesmí být vykonávána. Pokud je prvek uveden bez barvy, znamená to že má být vyhodnocován (proveden) standardním způsobem.

Struktura odpovídající zvolení konkrétního modulu v navigačním stromu je uvedena na obrázku 10.

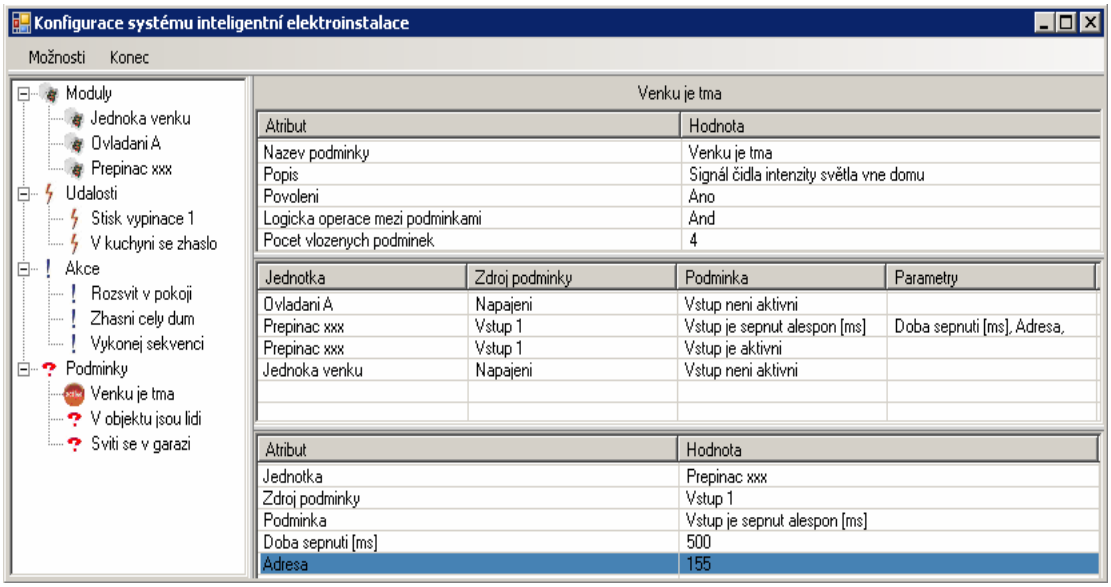
Obrázek 10:



Zobrazeny jsou všechny parametry zvoleného modulu rozdělené do záložek podle skupin příbuznosti definovaných v konfiguračním souboru tagem *tab.* Parametry jsou podbarveny v závislosti na hodnotě tagu *style* v jejich definici, což zvyšuje přehlednost seznamů. Změny hodnot jsou prováděny standardním výše zmíněným způsobem, přičemž u názvu jednotky a adresy je kontrolováno zachování jednoznačnosti (pokud je zadán již existující název či adresa, je text zobrazen červeně a případné potvrzení vede k ignorování změny).

Struktura odpovídající zvolení konkrétní podmínky v navigačním stromu je uvedena na obrázku 11.

Obrázek 11:



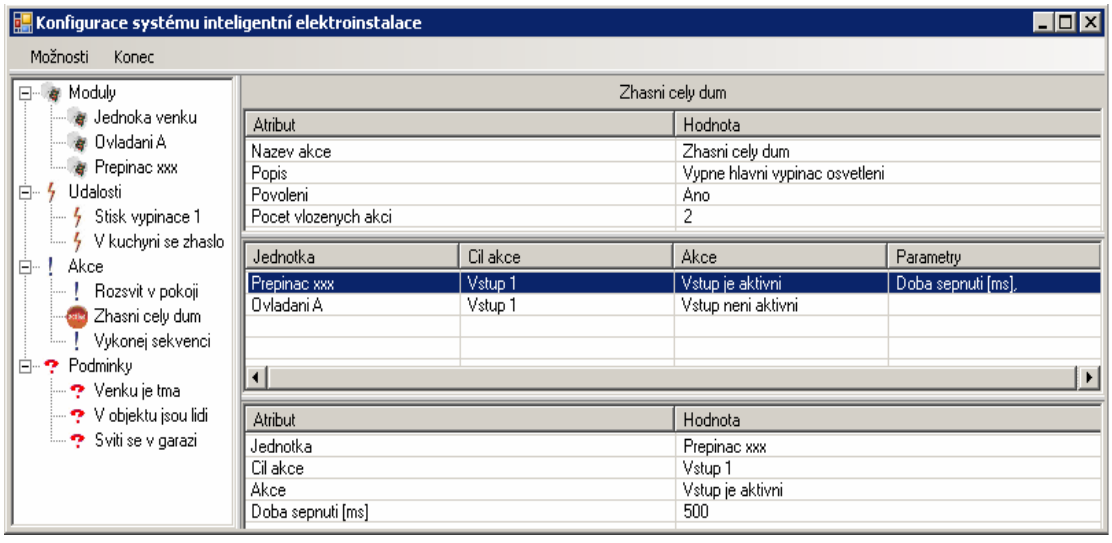
V první tabulce jsou uvedeny obecné parametry konkrétní podmínky jak byly uvedeny v popisu konfiguračního souboru, a navíc ještě počet vložených podpodmínek. Při změně názvu podmínky je již uvedeným způsobem kontrolováno dodržení jednoznačnosti jmen.

Druhá tabulka obsahuje soupis podpodmínek, jež jsou součástí této podmínky. Tyto podpodmínky jsou přidávány a odebírány výběrem příkazu v kontextovém menu, které je vyvoláno stiskem pravého tlačítka myši v prostoru tabulky (pokud je na pozici stisku nějaká podpodmínka je nabídnuto její odebrání, pokud ne, je nabídnuto přidání nové podpodmínky). Po volbě přidání podpodmínky je okamžitě do seznamu přidána podpodmínka s hodnotami všech atributů nastavenými na „?“. Hodnoty dále určuje uživatel v následující tabulce.

Poslední tabulka obsahuje definici konkrétní podpodmínky, konkrétně té, která byla vybrána v tabulce výše. Pokud nebyl žádný výběr uskutečněn, pak je tato tabulka uživateli nepřístupná. V této tabulce dochází k definici podpodmínek. Aby bylo možné nabízet uživateli vždy požadovaná data a přitom zachovat univerzálnost systému, má definice podpodmínek přesný postup, přičemž platí, že další krok postupu je povolen až po úspěšném provedení toho aktuálního. V první řadě musí uživatel zvolit, který z modulů bude pro tuto podpodmínku zdrojový (při změně parametru Modul jsou v comboboxu nabízeny názvy modulů, které jsou v systému definovány). Druhým krokem je pak určit parametr, jehož hodnota bude v podmínce vyhodnocována (při změně parametru zdroj podmínky je v comboboxu uveden seznam parametrů modulu s názvem definovaným v prvním kroku, které mohou sloužit k vyhodnocení podmínek). V třetím a posledním kroku pak uživatel definuje která z podmínek, jež mohou být vyhodnocovány nad zvoleným parametrem má být opravdu použita (při změně parametru Podmínka je v comboboxu uveden seznam podmínek, jež mohou být nad výše definovaným parametrem vyhodnocovány). Po úspěšném zadání všech tří parametrů jsou do tabulky přidány případné další parametry potřebné k vyhodnocení nadefinované podpodmínky a je umožněna jejich modifikace.

Pokud chce uživatel provést změny ve vlastnostech již definované podmínky, jsou parametry uvedené v postupu za měněným parametrem smazány a musí být definovány znovu. Struktura odpovídající zvolení konkrétní akce v navigačním stromu je uvedena na obrázku 12.

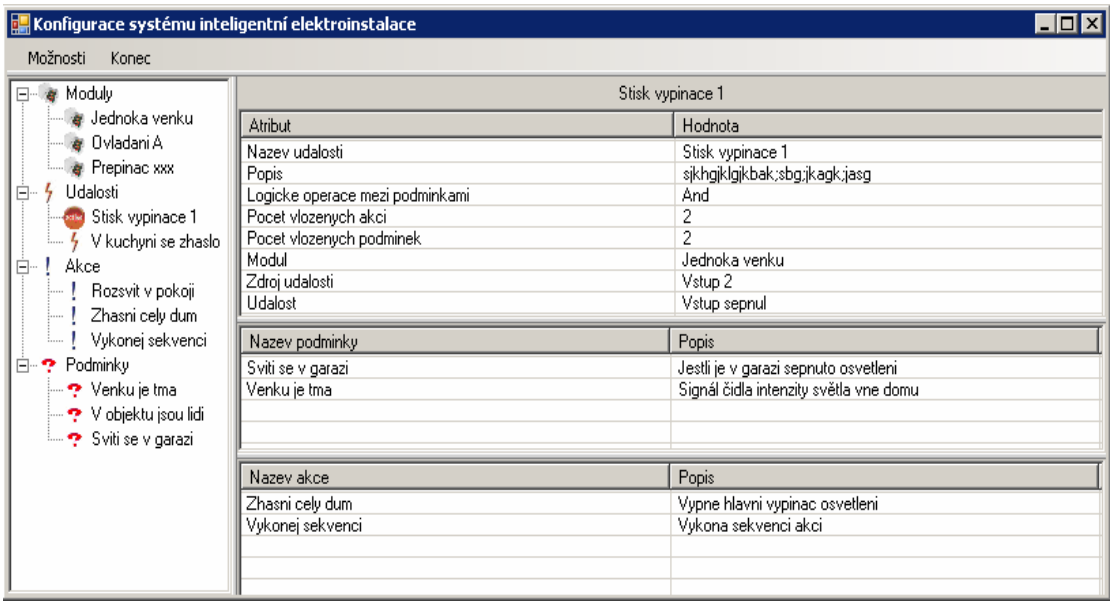
Obrázek 12:



Funkce tabulek v této situaci jsou naprosto analogické s funkcemi pro podmínky a proto nebudou detailně popisovány.

Struktura odpovídající zvolení konkrétní události v navigačním stromu je uvedena na obrázku 13.

Obrázek 13:



V první tabulce jsou uvedeny obecné parametry události, které není třeba dále popisovat a definice určité změny konkrétního parametru, který spouští vyhodnocení podmínek a případné provedení akcí. Tato změna je definována pomocí parametrů Modul, Zdroj události a Událost a to stejným postupem, jaký byl popsán pro definici podpodmínek. Pokud má takto definovaná změna ještě další parametry, pak jsou uvedeny na konec tabulky a je umožněna jejich editace.

Druhá tabulka obsahuje seznam podmínek, které mají být v rámci provedení této události vyhodnoceny. Ty jsou přidávány pomocí kontextového menu, které je vyvoláno stiskem pravého tlačítka myši v prostoru tabulky (pokud je na pozici stisku nějaká podmínka je nabídnuto její odebrání, pokud ne je nabídnuto přidání nové podmínky). Po volbě přidání nové podmínky zvolí uživatel požadovanou podmínku z jím dříve definovaných podmínek (jsou nabídnuty jako hodnoty comboboxu). V této tabulce není možné měnit hodnoty zobrazovaných atributů (název a popis podmínky).

Třetí tabulka obsahuje seznam akcí, které se mají při kladném vyhodnocení podmínek v rámci události vykonat. Její funkce jsou analogické s předchozí tabulkou.

3.4 Služební část konfiguračního softwaru

Ke komunikaci s řídící jednotkou je využíván běžný osobní počítač, vybavený komunikačním portem pro sériovou linku, případně přístupem k internetu. Vzhledem k nízké výpočetní složitosti úloh požadovaných pro provoz konfiguračního softwaru (příjem dat a jejich odeslání na sériovou linku) je jistě zbytečné, aby byl pro tyto účely vyhrazen samostatný, byť nevýkonný, počítač a tyto funkce lze svěřit osobnímu počítači, který je využíván i pro jiné aplikace. V ideálním případě by měly úlohy související s činností konfiguračního systému běžet zcela na pozadí tak, aby se o ně běžný uživatel po jejich nainstalování nemusel vůbec starat.

Takového efektu lze dosáhnout tím, že tyto úlohy implementujeme tak, aby běžely nezávisle na uživatelských kontech v rámci operačního systému. Proto byla implementována aplikace typu windows service, která je spouštěna při inicializaci operačního systému Microsoft Windows a je aktivní po celou dobu jeho činnosti, nezávisle na užívaných uživatelských kontech a jiných aplikacích. Udržovat zapnutý obslužný osobní počítač v době kdy má být umožněna změna konfigurace řídící jednotky, jejíž řídící činnost je ale samozřejmě na aktivitě tohoto počítače nezávislá je tedy jedinou podmínkou, kterou je nutné splnit aby bylo možné změny provádět.

Činnost této části sestává z naslouchání na portu TCP/IP číslo 12 001 a čekání na data od klienta. Pokud se na tomto portu objeví data, jsou vyhodnocena jako heslo a toto je porovnáno s heslem požadovaným pro autorizaci přístupu k řídící jednotce. Pokud je heslo akceptováno, je potvrzeno spojení s klientem a další data jsou interpretována jako obsah konfiguračního souboru a ukládána do souboru na místním disku. Po obdržení kompletního souboru, což je signalizováno speciální datovou sekvencí, je ukončeno spojení s klientem a konfigurační soubor je odeslán po sériové lince do řídící jednotky.

4 Závěrečné zhodnocení a návrhy pro další rozšíření

V souladu se zadáním byl navržen software sloužící ke konfiguraci funkcí systému inteligentní elektroinstalace. Software přitom pracuje pouze s funkcemi, jejichž vykonání je podmíněno změnou jednoho z parametrů jedné z jednotek systému. Díky užití principu definičních souborů a specifikace funkcí systému pomocí různých kombinací základních prvků, popisujících elementární funkcionalitu, bylo

docíleno vysokého stupně obecnosti konfiguračního softwaru a nezávislosti na hardwarovém složení konfigurovaného systému. Pro zvýšení uživatelského komfortu a přehlednosti bylo do programu zahrnuto i grafické rozhraní, s jehož pomocí lze snadno provádět změny v existující konfiguraci, nebo vytvářet konfigurace nové.

V případě dalšího vývoje aplikace, se nabízí několik směrů pro rozšíření funkcionality. Velmi přínosným rozšířením by jistě bylo zahrnutí časových funkcí, s jejichž pomocí by bylo možné snadněji definovat například funkce jež mají být aktivní jen v určitý čas či den. Dalším možným rozšířením by pak mohla být samostatná část softwaru sloužící k online monitoringu stavu jednotlivých spotřebičů, nejlépe v grafické podobě ve formě tzv. floorplanu, kdy jsou jednotlivé spotřebiče a jejich stav reprezentovány měnícími se ikonami na plánu vnitřních prostor budovy, přičemž umístění ikon na plánu odpovídá skutečnému umístění spotřebičů v reálné budově.

Seznam použité literatury a softwaru

Začínáme programovat v C#, Eric Gunnerson, Computer Press, Praha, 2001
www.wikipedia.org
Microsoft Visual Studio 2005