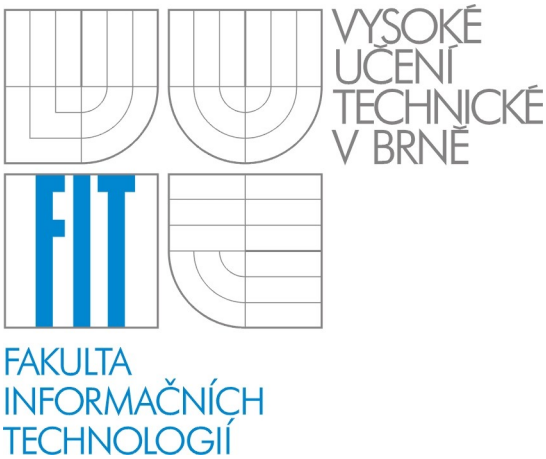
Vysoké učení technické v Brně
Fakulta informačních technologií



Visual1451: .NET implementace prostředí pro IEEE1451 aplikace

Diplomová práce

(semestrální část)

2007 Jiří Macháček

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ondřeje Ryšavého, z Ústavu informačních systémů na Fakultě informačních technologií Vysokého učení technického. Uvedl jsem všechny literární prameny a publikace ze kterých jsem čerpal.

V Brně Jiří Macháček

Abstract

The family of IEEE1451 standards deals with application and communication architecture of embedded systems interconnected in sensor networks. We provide a general elaboration of the standard, establishing the foundation for further more detailed analysis and implementation by means of software components of Microsoft .NET framework. This report mainly focuses at specification of object, data and communication models of the standard. Selected parts of application architecture are modeled by UML.

Key Words

IEEE1451 standard, sensor networks, embedded systems, object model, network communication, .NET framework

Abstrakt

Standard IEEE1451.1 popisuje aplikační a komunikační architekturu věstavěných systémů tvořící senzorové sítě. Tato práce se zabývá jeho zevrubnou analýzou sloužící jako základ pro následnou analýzu detailní a implementaci ve formě softwarového prostředí pro aplikační doménu Microsoft .NET. V práci je popsán a rozebrán objektový, datový a komunikační model. Vybrané části aplikační architektury jou modelovány jazykem UML.

Klíčová slova

IEEE1451 standard, senzorové sítě, vestavěné systémy, objektový model, síťová komunikace, .NET framework

Obsah

1 Úvod....6
2 Standard IEEE1451....8
2.1 Vznik a účel standardu IEEE1451....8
2.2 Pohledy na standard....8
2.3 Uživatelé standardu IEEE1451....10
2.4 Objektový model....10
2.5 Datový model....13
2.6 Síťový komunikační model....13
2.7 Model správy paměti....15
3 Návrh IEEE1451 prostředí....16
3.1 UML návrh tříd....16
3.2 Stavový popis....19
4 Závěr....24

1 Úvod

Tato práce se zabývá popisem a dekompozicí standardu IEEE1451. Tento standard se obecně zabývá popisem, skládáním a spojováním senzorových sítí [2]. Vznikl proto, aby se dalo lépe nahrazovat a upravovat existující infrastrukturu a případně spojovat různé sítě a jednoduše dopomáhat dalšímu růstu. Toto by bylo bez jednotného standardu velmi obtížné a hlavně finančně a časově poměrně náročné [3].

Standard definuje především převod z vlastních senzorů na nějaký typ přenosové sítě. Další důležitá věc je, že na síti může být více zařízení, které spolu mohou různě komunikovat. A to ať už formou klient-server, tak formou many-to-many. Vše je definováno plně objektově tedy i komunikace probíhá jako volání objektů na jiných místech sítě. Komunikace probíhá pomocí toho, že tyto objekty jsou nastaveny jako síťově viditelné. Aby toto mohlo být plně kompatibilní, tak musí být ve standardu plně definovány i všechny použitelné datové typy, aby nedošlo k pomíchání typů. S tímto také souvisí správa paměti ta je obdobně jako v jazyce C založena na tom, že o alokace a dealokace se musí programátor starat sám a není plně automatická jako například u programovacího jazyku C# [4] a podobných moderních jazyků.

Hlavní dva pohledy na tento standard jsou logický a fyzický, kdy fyzický se zabývá vlastním zapojením, hardwarem, fyzickými komponenty a podobně. Zato logický pohled je o komunikačních protokolech a abstrakci hardwaru a podobných věcech a je implementován ve firmwaru, softwaru případně ve službách operačního systému. Tyto dva pohledy se vzájemně doplňují a tvoří jeden celek.

Dokument je rozdělen na čtyři základní části. První byl úvod, ve kterém byl čtenář seznámen převážně s tím, proč standard vznikl a k čemu slouží a s jeho základními vlastnostmi a parametry.

Druhou kapitolu tvoří vlastní popis IEEE1451, něco o jeho užití a kdo jej používá a proč. V další podkapitole je rozebráno jaký používá objektový model, tedy jak na sobě vzájemně jednotlivé objekty závisí. A každá část modelu je popsána jaký je její význam. Následuje popis datového modelu tam je rozebráno jaké jsou používány datové typy a podobně, toto je důležité, kvůli kompatibilitě při síťové komunikaci, aby byly pevně definovány použité datové typy. Následuje popis toho jak probíhá vlastní síťová

komunikace v sítích IEEE1451. Komunikace v IEEE1451 mohou být dvou typů klient-server a many-to-many v této kapitole bude obojí popsáno. A nakonec druhé kapitoly je popsáno jaká je správa paměti, jak je řešena a za co je odpovědný programátor a co se řeší automaticky.

Ve třetí kapitole je popsána dekompozice do jednotlivých tříd a vlastní návrh prostředí IEEE1451, obsahuje především UML [5] diagramy tříd, jež popisují jak na sobě jednotlivé třídy závisí. Dále se v této kapitole nachází několik stavových diagramů, jež popisují stavové řízení v několika vybraných zajímavých případech. Z důvodu rozsáhlosti standardu nemohou být v práci uvedeny všechny diagramy, tak jsme vybral ty podle mého názoru nejvíce ilustrativní.

Práce je zakončena závěrem, který diskutuje další pokračovaní projektu jenž bude zakončen diplomovou prací.

2 Standard IEEE1451

V této kapitole bude popsán standard IEEE1451 a to hlavně, proč vznikl, kdo je organizace IEEE. Jaké jsou dva pohledy na tento standard. Dále, kdo tento standard používá a k čemu. A zakončena bude popisem objektového modelu tohoto standardu jak je navržen.

2.1 Vznik a účel standardu IEEE1451

Jak již název napovídá tento standard navrhla skupina IEEE tedy Institute of Electrical and Electronics Engineers. Tato společnost, jež se zabývá návrhem různých standardů pro elektroniku z těch nejznámějších bych jmenoval, například Ethernet, WiFi, Bluetooth, WiMax, USB, paralelní port, FireWire, JTAG, toto je jenom malá část z více než 900 standardů, které IEEE vytvořilo. Tato organizace se zabývá především vytvářením a prosazováním standardů v oblasti elektřiny, elektrotechniky, komunikací a počítačových oborů. IEEE publikuje k těmto tématům 30% veškeré světové vědecké literatury.

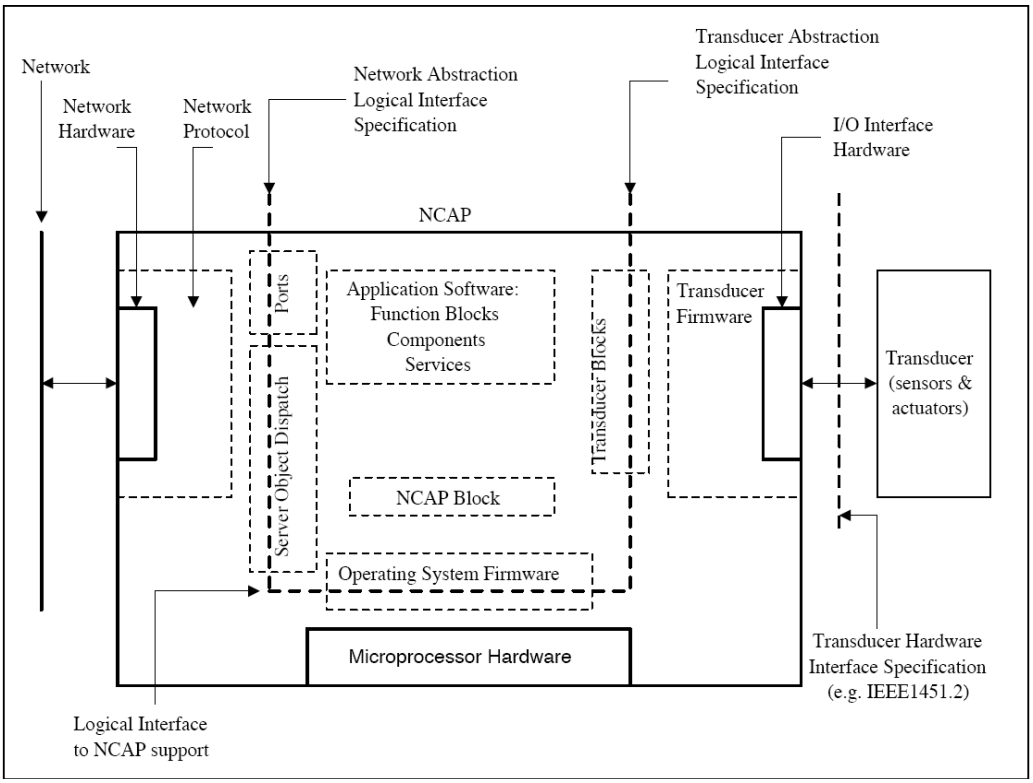
Standard IEEE1451 vznikl za účelem použit existující kontrolní síť a vytvořit k ní standardizovanou metodu pro jejich spojování, řízení a další rozšiřování. Protože existovalo spoustu vzájemně nekompatibilních řešení, tak se objevovalo mnoho různých problému, ať již s nahrazováním části infrastruktury novější nebo propojováním této infrastruktury s jinou a podobně. Díky tomuto standardu poté stačí např. při změně zařízení v propojení udělat pouze malé nebo dokonce žádné změny v zapojení a infrastruktuře a síť může prakticky okamžitě pracovat dále. Tento standard tedy zkracuje čas pro různé rozšiřování údržby a podobně.

2.2 Pohledy na standard

Hlavním základem IEEE1451 je tzv. NCAP je to zkratka od Network Capable Application Processor. Tento procesor je takový inteligentní most mezi senzorem a sítí, po které se přenáší data a případně jsou na ni připojeny další NCAP. Když je na síti více

NCAP, tak spolu mohou vzájemně komunikovat, toto bude popsáno v další podkapitole. NCAP jako takový je obvykle složen z mikroprocesoru a další podpůrné logiky.

Tato technologie je založena na dvou hlavních pohledech na tuto síť a to pohledu fyzickém a pohledu logickém viz následující obrázek 1.



Obrázek 1. fyzický a logický pohled

Fyzický pohled popisuje především připojení senzorů a čidel k hlavnímu mikroprocesoru, fyzické propojení se sítí, různé porty a podobně. Na obrázku je toto znázorněno plnou čarou. Je zde vidět připojení senzoru přes rozhraní přes mikroprocesor, jež dále zprostředkovává spojení se sítí a dále přenáší signály ze senzoru do další infrastruktury.

Logický pohled je součásti firmware nebo služeb operačního systému a podobně. Jedná se o různé služby, síťové protokoly, abstraktní pohled na hardware. Ten je znázorněn přerušovanou čarou. Z tohoto pohledu je tu znázorněno jak firmware vlastního

převodníku vlastní přenosové sběrnice uvnitř řídící jednotky a nakonec síťový protokol, jež zajišťuje logiku komunikace dále do sítě.

2.3 Uživatelé standardu IEEE1451

Tento standard je v praxi primárně určen pro 3 typy uživatelů, kteří s ním pracují a nebo pro něj vytváří různé funkcionality a podobně. Standard používají:

1. Systémový vývojáři

   Zabývají se vlastním návrhem systémů pro IEEE 1451 a tyto hlavně návrhem potřebných tříd a knihoven zabývajících se infrastrukturou sítě.

2. Vývojáři komponent

   Tito navrhují znovu použitelné komponenty pro systémy, jež jsou ještě dále používány v jiných částech sítě či jiných projektech.

3. Koncový uživatelé

   Sestavují a instalují tyto navržené sítě do daného prostředí, kde budou nadále pracovat.

2.4 Objektový model

Objektový model IEEE1451 je skládán ze stavebních bloků pro aplikační systémy. Standardní specifikace každé objektové třídy definují dvě základní věci a to rozhraní třídy vyjadřujících popisy objektových operací. A druhou je čistě textový popis a odpovídající automat ukazující jeho chování.

Na následujícím obrázku 2 je ukázán základní koncept objektového modelu. Na obrázku je vidět jak spolu jednotlivé třídy souvisí, zanořují se do sebe a jak jsou rozděleny. Kromě jmen tříd jsou zde zobrazeny i jejich ID, což je ono číslo, které lépe usnadňuje orientaci.

Entity 1.1
- *Block* 1.1.1
  - **NCAP Block** 1.1.1.1
  - *Function Block* 1.1.1.2
  - *Base Transducer Block* 1.1.1.3
    - *Transducer Block* 1.1.1.3.1
      - **Dot2 Transducer Block** 1.1.1.3.1.1
      - **Dot3 Transducer Block** 1.1.1.3.1.2
      - **Dot4 Transducer Block** 1.1.1.3.1.3
- *Component* 1.1.2
  - **Parameter** 1.1.2.1
    - **Parameter With Update** 1.1.2.1.1
      - *Physical Parameter* 1.1.2.1.1.1
        - **Scalar Parameter** 1.1.2.1.1.1.1
          - **Scalar Series Parameter** 1.1.2.1.1.1.1.1
        - **Vector Parameter** 1.1.2.1.1.1.2
          - **Vector Series Parameter** 1.1.2.1.1.1.2.1
    - **Time Parameter** 1.1.2.1.2
  - **Action** 1.1.2.2
  - **File** 1.1.2.3
    - **Partitioned File** 1.1.2.3.1
  - **Component Group** 1.1.2.4
- *Service* 1.1.3
  - *Base Port* 1.1.3.1
    - *Base Client Port* 1.1.3.1.1
      - **Client Port** 1.1.3.1.1.1
      - **Asynchronous Client Port** 1.1.3.1.1.2
    - *Base Publisher Port* 1.1.3.1.2
      - **Publisher Port** 1.1.3.1.2.1
      - **Self Identifying Publisher Port** 1.1.3.1.2.2
        - **Event Generator Publisher Port** 1.1.3.1.2.2.1
  - **Subscriber Port** 1.1.3.2
  - **Mutex Service** 1.1.3.3
  - **Condition Variable Service** 1.1.3.4

Obrázek 2. Hierarchie tříd v IEEE1451

Na obrázku je vidět, že se zde nachází hlavní 4 objektové třídy.

- Blokové třídy

  IEEE1451_NCAPBlock class – poskytuje standardní softwarové rozhraní pro podporu síťové komunikace a systémovou konfigurací.

  IEEE1451_BaseTransducerBlock class – poskytuje standardní softwarové rozhraní mezi převodníky a aplikačními funkcemi.

  IEEE1451_FunctionBlock class – zapouzdří aplikačně specifickou funkcionalitu.

- Třídy komponenty

  - Strukturované informace například měření a soubory.

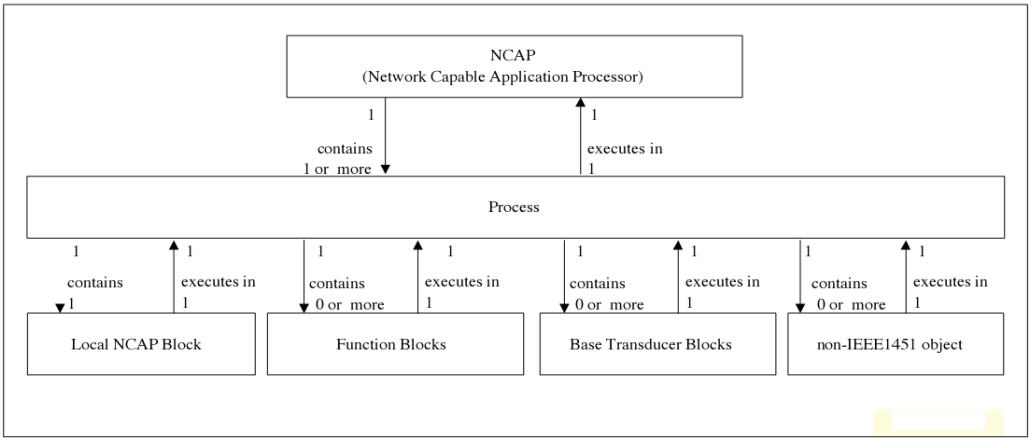
  - Kolekce odpovídajících aplikačně specifických objektů.

- Akce se stavem, v případě, že tato akce trvá relativně dlouhou dlouhou dobu.

- Servisní třídy
  - Komunikace mezi objekty na různých NCAP.
  - Systémové synchronizace.

- Jiné než IEEE1451 objekty
  - Zde jsou umístěny ostatní objekty, jež chceme volat v NCAP síti a nejsou obsaženy ve standardu.

Ve standardu je popsáno, jak na sobě jednotlivé objekty závisí. Následuje obrázek 3 ukazuje, jak výše popsané třídy závisí na hlavním NCAP, některé vazby jsou binární a některé unární.



Obrázek 3. Architektura objektového modelu

Je zde vidět, že například jeden procesor musí obsahovat, alespoň jeden proces, ale může provozovat i více procesů, ale proces může běžet pouze na jednom procesoru.

Třídy mohou být i síťově viditelné, ale musí proto splnit následující tři podmínky. Třída musí být podtřídou IEEE1451_Entity a zároveň musí být registrována v jejím lokálním NCAP blokem. Poslední podmínkou je že musí být vlastněna lokálním NCAP blokem.

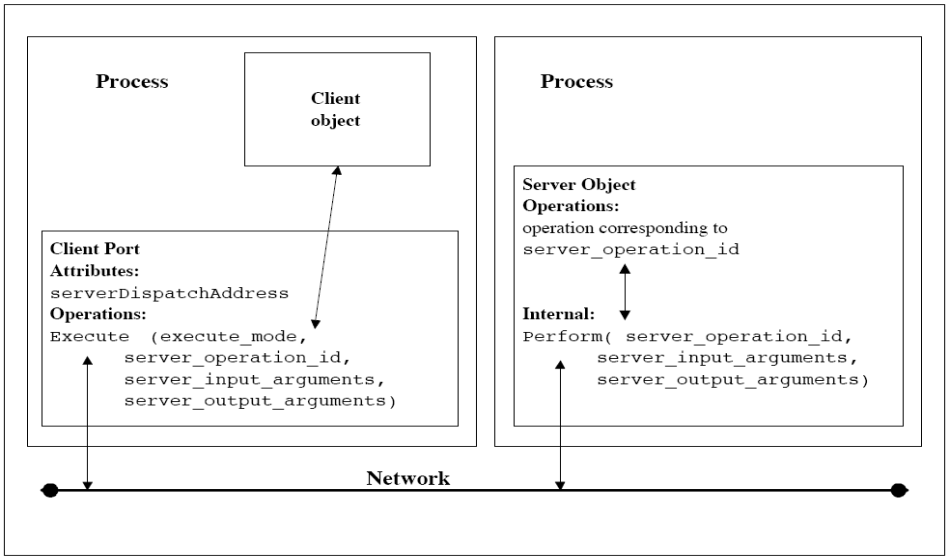
2.5 Datový model

Jeho úkolem je hlavně specifikovat typy a formu komunikace přes komunikační síť v rámci standardu IEEE1451 a to jak komunikaci lokální, tak vzdálenou přes komunikační sít. Je zde přesně definován každý datový typ a každá struktura, jež je používána při komunikaci, aby nemohlo dojít k nechtěným různým definicím téhož typu a podobně. Základní datové typy jsou podobné jako v jazyce C jsou to Boolean, octet, Integery, Typy s plovoucí desetinou čárkou a řetězcové typy.

2.6 Síťový komunikační model

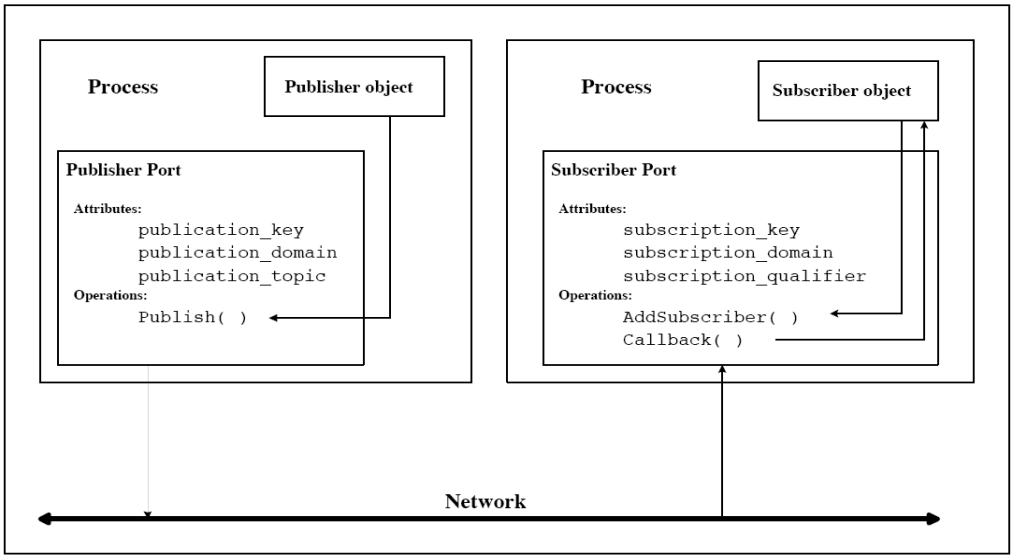
V IEEE1451 jsou definovány základní 2 druhy komunikací a to komunikace point-to-point, která je prováděna stylem klient-server, když se chtějí spojit spolu 2 body v síti. A druhou možností komunikace je komunikace one-to-many nebo many-to-many, kdy spolu vzájemně komunikuje více bodů v síti. Komunikační model definuje syntax a sémantiku softwarového rozhraní mezi aplikačními objekty a komunikační sítí. Toto je definováno abstraktně tedy plně nezávisle na specifikacích komunikačních sítí, jež jsou používány. To že nejsou ve standardu definovány komunikační protokoly znamená, že pro každé konkrétní použití v síti si musí koncový uživatel přidat vlastní knihovny, jež zprostředkovávají volání mezi IEEE1451 komunikačními operacemi a vlastní sítí. Tyto knihovny by též měly zajišťovat to, aby aplikační vývojáři byly ušetřeny řešení mnoha detailů, pomocí větší abstrakce síťového spojení. Tímto se ušetří jak čas při implementaci, tak se také zabrání vnášení nechtěných chyb do kódu a podobně.

Komunikace klient-server je založena na dvou vzájemně se doplňujících metodách. První je vykonávání nějakého objektu na straně klienta a druhou je provádění na všech síťově viditelných serverových objektech. Tyto dvě metody jsou taková obdoba zasílání zpráv viz obrázek 4.



Obrázek 4. Síťová komunikace klient-server

Druhou možností komunikace je komunikace mezi více uzly. Ta probíhá tak, že jeden vysílá objekty do sítě a ostatní je přijímají a provádějí. Komunikace probíhá na principu publikování a podepisování. Viz obrázek 5, na kterém je znázorněn jeden proces, který odešle přes síť svoje data a jeden z procesů na síti jej přijme. Těchto procesů, které zprávu přijímají bývá většinou více.



Obrázek 5. Síťová komunikace klient-server

2.7 Model správy paměti

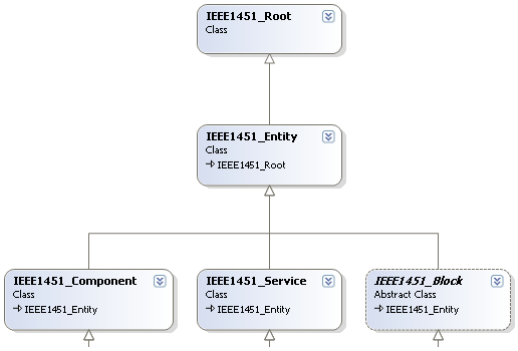
Implementace IEEE1451 je založena na tom, že za alokaci a dealokaci neprimitivních datových typů je odpovědný program a ne programovací jazyk. Je to tedy podobně jako v jazyce C a nikoliv jako v novějším C#, kde je vše řešeno automaticky za cenu jistého dopadu na výkonnost. Díky tomu bývá v jazyce C výkonnost obecně větší, než v novějších jazycích jako C# a JAVA. U primitivních datových typů viz kapitola 2.2 je za jejich alokaci a dealokaci odpovědný blok kódu, kde jsou deklarovány. U složitějších typů, jež se používají při komunikaci po síti je to tak, že za jejich spravování jsou odpovědné komponenty, jež se zúčastňují této komunikace.

3 Návrh IEEE1451 prostředí

V této kapitole se rozebere, jak je vytvářen objektový návrh implementace IEEE1451. Budou zde ukázány UML diagramy, aby bylo názorně vidět jak jsou třídy strukturovány. A několik ukázek stavových diagramů a to těch nejzajímavějších, protože norma je velice rozsáhlá a na všechny by nebylo místo.

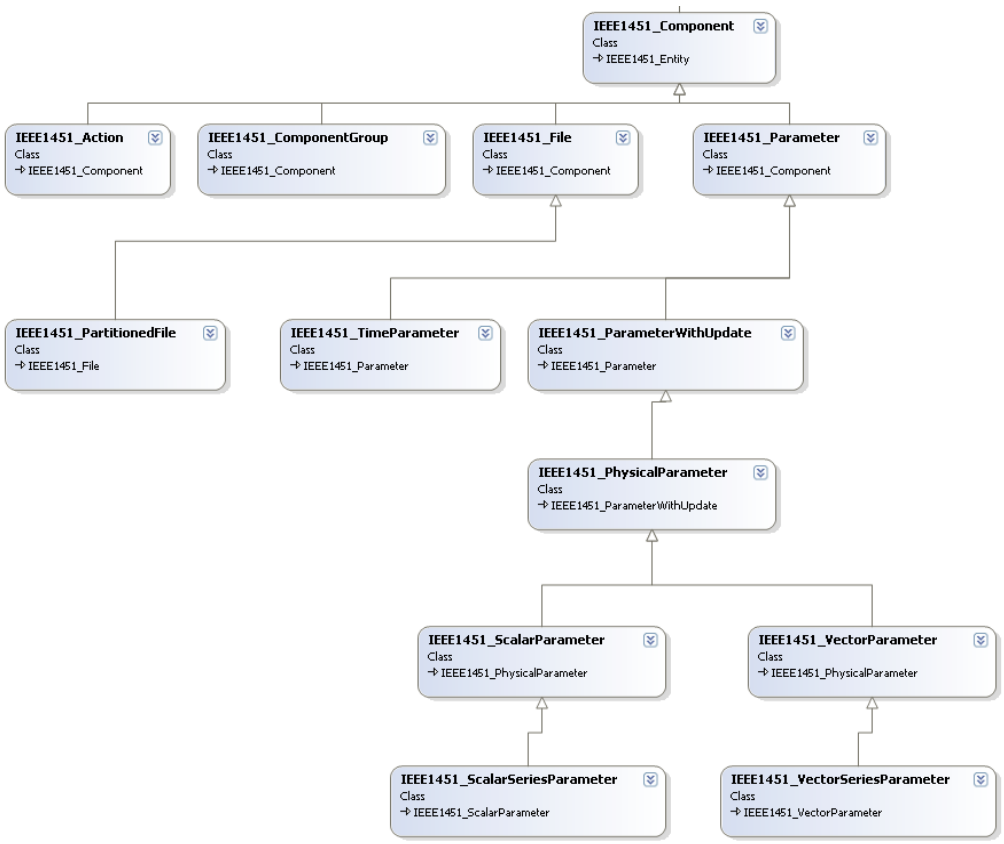
3.1 UML návrh tříd

UML diagram tříd jsem pro větší přehlednost rozdělil do pěti částí viz níže. Ke každé části bude přiřazen popis, co obsahuje.

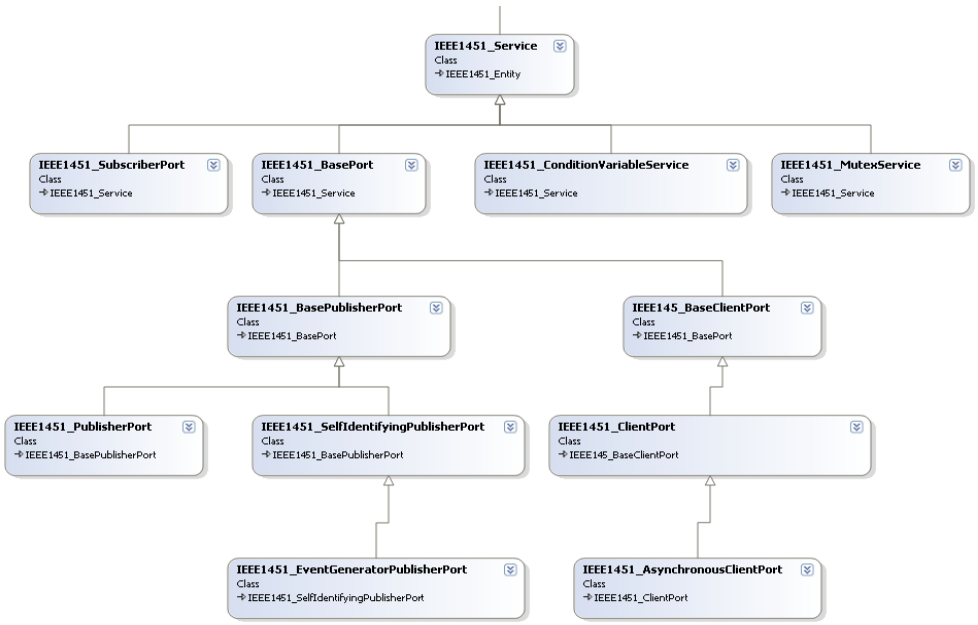


Obrázek 6. UML diagram základní rozdělení tříd

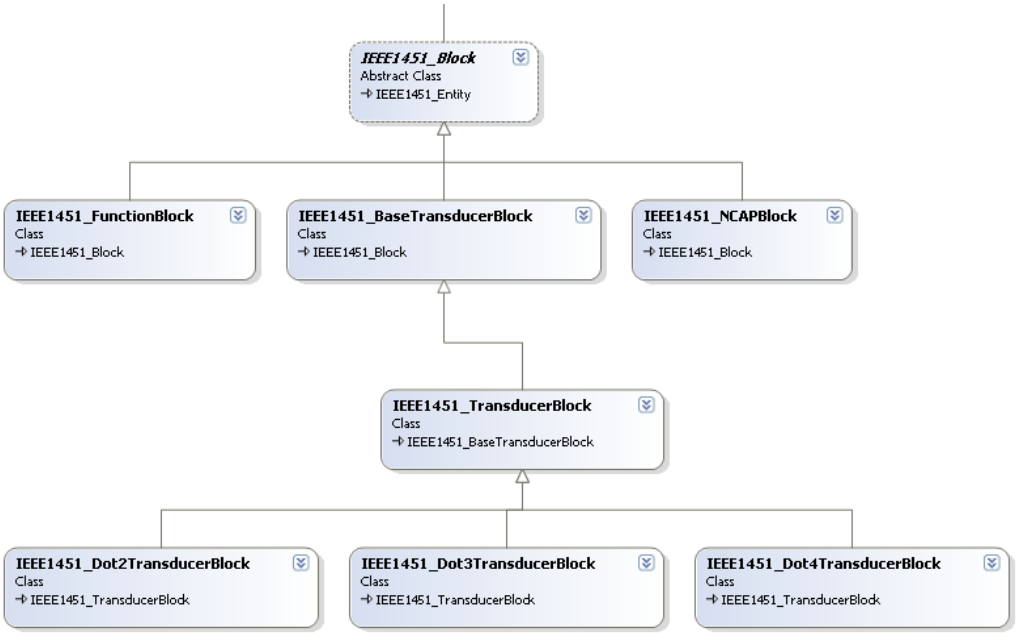
Na obrázku 6 je vidět kořenová třída, na které potom závisí další třídy. Kvůli rozsáhlosti tohoto diagramu jsem jej rozdělil na další tři poddiagramy viz níže. Tyto tři diagramy jsou pokračováním tříd Component, Service a Block.



Obrázek 7. UML diagram pro podtřídu Component



Obrázek 8. UML diagram pro podtřídu Service



Obrázek 9. UML diagram pro podtřídu Block

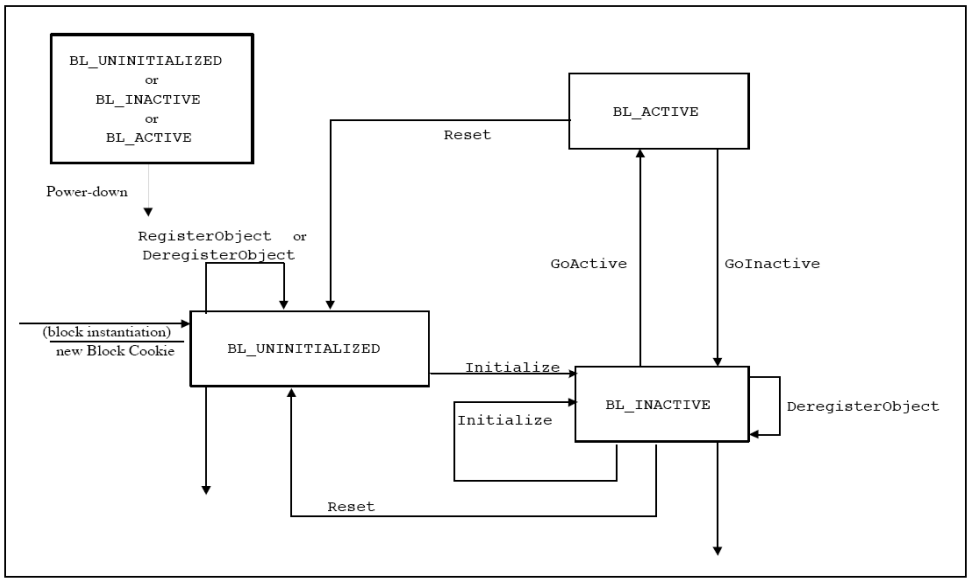
Obrázek 10. UML diagram dalších pomocných tříd

Na těchto UML diagramech je v ukázáno, jak jsou jednotlivé třídy strukturovány a jaké jsou jejich vnitřní vazby a jaké jednotlivé třídy je třeba implementovat. Na posledním obrázku 10 jsou ukázány pomocné třídy, jež nejsou přímo spojeny s ostatními, ale jsou požity například pro signalizaci chyb a podobně.

3.2 Stavový popis

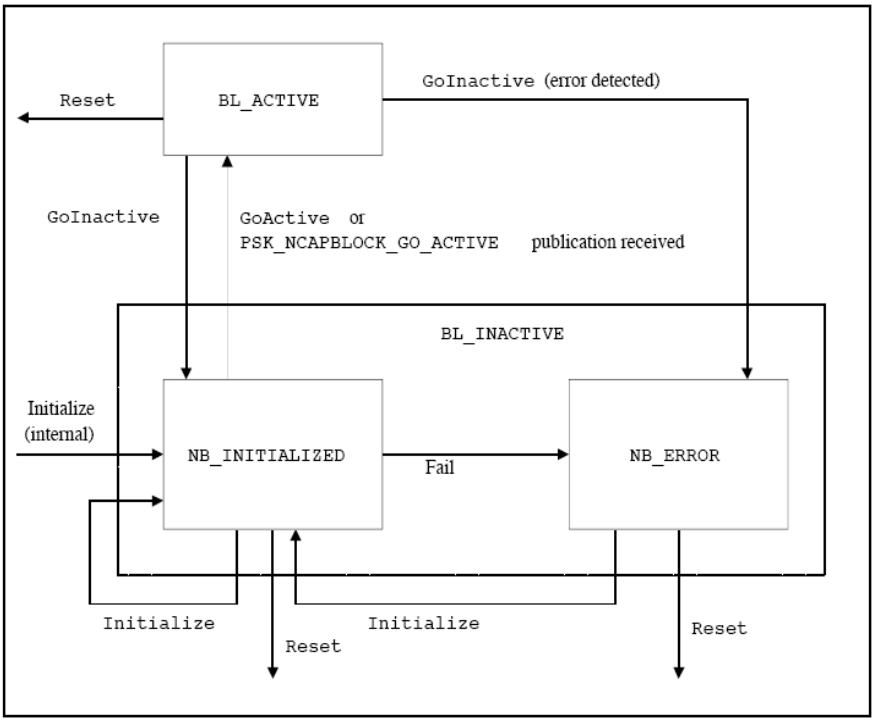
V této kapitole se popíše, jak jednotlivé třídy jsou popsány stavovými automaty. Jakých stavů vnitřně dosahují, do jakých stavů poté při nějaké interakci přecházejí a podobně.

Na následujicím obrázku je modelováno chování Block třídy popisující vlastní NCAP procesor.



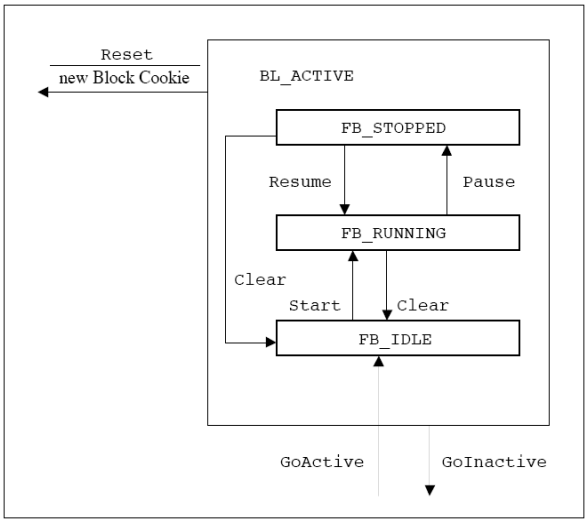
Obrázek 11. Konečný automat reprezentujicí třídu Block a její životní cyklus

Na obrázku 11 je modelovan Block a jeho životní cyklus a je zde vidět počáteční stav BL_UNINITIALIZED do tohoto stavu se přejde po zapnutí, výpadku napájení nebo resetu. Z tohoto stavu můžeme přejít do stavu BL_INACTIVE, tento stav znamená, že je blok inicializován, ale není aktivní. Blok se může aktivovat a přejít do stavu BL_ACTIVE, kdy je aktivní a provádí nějakou činnost, poté se může opět buď deaktivovat nebo přes reset může přejít zpět do počátečního stavu.



Obrázek 12. NCAP Block

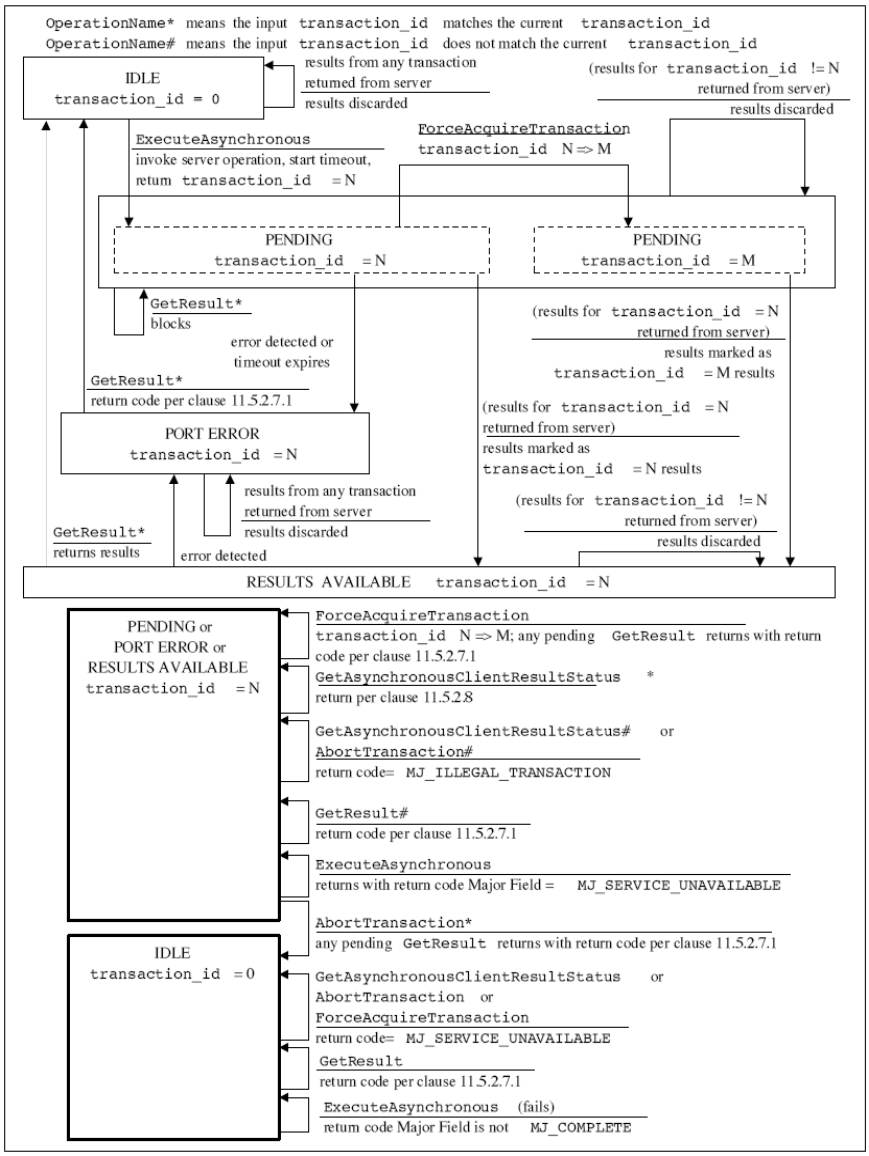
Obrázek 12 ukazuje chování NCAP Blocku, jedná se o obdobné chování jako u předchozího modelu, jen je zde navíc doplněn chybový stav NB_ERROR. Je opět vidět, jak mezi sebou vzájemně stavy přecházejí při různých podnětech.



Obrázek 13. Stavový popis pro Function Block

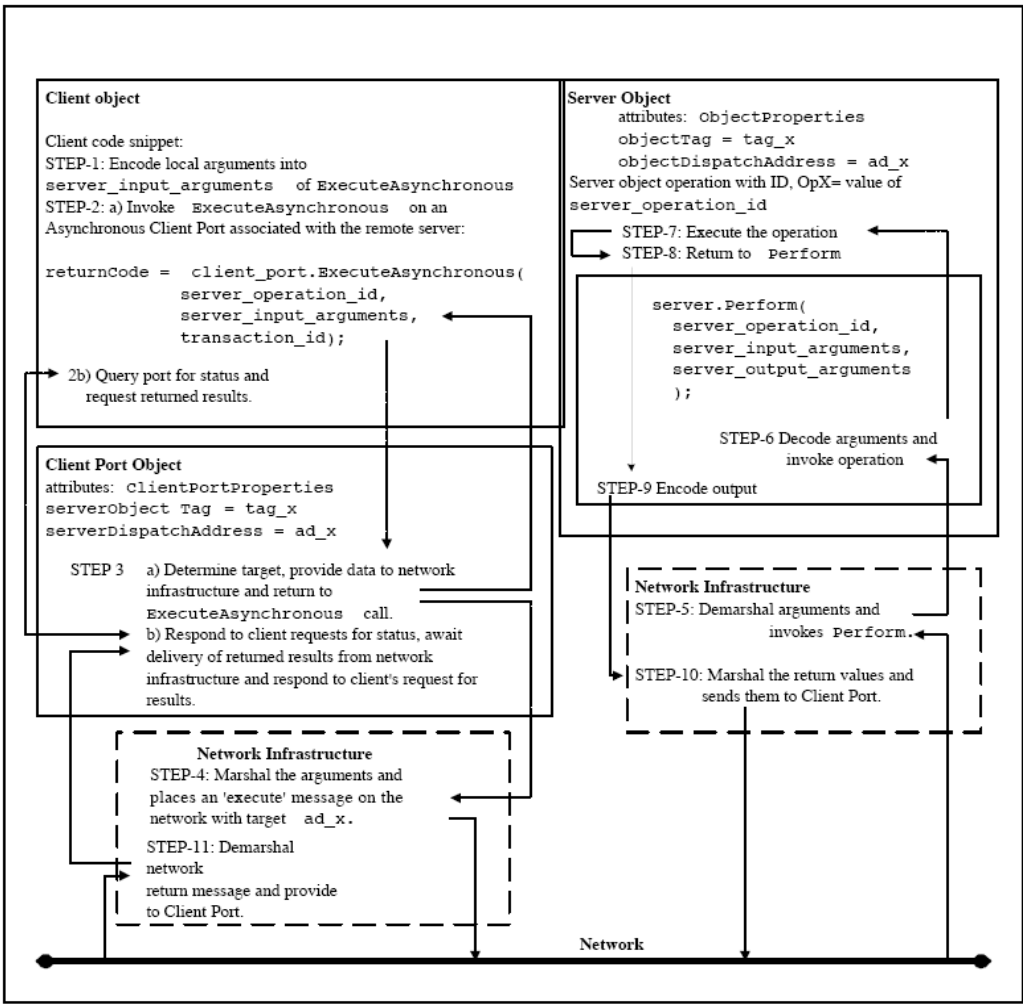
Obrázek 13 ilustruje jaká je vnitřní funkcionalita ve Function Block. V tomto bloku se spustí proces, který může být pozastavován a opět spouštěn a nebo případně úplně zrušen a poté znovu spuštěn. Případně může být kompletně ukončen přes reset nebo přes GoInactive.

Dále bude následovat několik obrázků s komentářem k reprezentující síťovou komunikaci.



Obrázek 14. Stavový automat pro Asynchronní klientský port

Reprezentuje stavový diagram komunikace a na následujicím obrázku 14 bude reprezentován jak tato komunikace postupně po jednotlivých krocích probíhá.



Obrázek 15. Asynchronní komunikace klient-server

Na přechozím obrázku je ilustrována komunikace klient-server, tak že je rozdělena do jednotlivých kroků, které je třeba udělat při komunikaci. A to včetně všech možných přechodů , které mezi těmito stavy mohou nastat. Od kroku jedna, kdy si klient připravuje data do formátu pro zpracování na serveru, poté pokračujeme do kroku dva, kde zařídíme odeslání dat po síti. Potom v dalším kroku číslo tři zahájíme čekání na odpověď ze strany serveru a v kroku čtyři odešleme veškerá data po síti. Na druhé straně je vidět jak server přijme požadavek a v kroku pět jej dekoduje. V následujicím kroku šest dekoduje dodané parametry. Následuje krok sedm, který provede zadaný požadavek a potom přes krok osm vrátí řízení zpět. V kroku devět zakódujeme výstupní data a připravíme je tak na odeslání po síti. V kroku deset tato data odešleme po síti na klientský

port. A klient je v kroku jedenáct přijme a tak je tato komunikace klient-server dokončena a může být případně opakován další požadavek a podobně.

4 Závěr

Tento dokument obsahuje úvodní analýzu aplikační domény sensorových sítí dle standardu IEEE1451.1. Získané znalosti slouží v dalších krocích pro detailní návrh a následnou implementaci v prostředí Microsoft .NET. Cílem je vytvořit prostředí pro návrh aplikací sensorových sítí. Toto prostředí nejen že umožní platformě nezávislý přístup k návrhu a implementaci aplikací, ale mělo by se stát i hodnotným experimentálním prostředím pro vývoj a realizaci specializovaných vývojových nástrojů, jako jsou generátory kódů z doménově specifických modelovacích jazyků, či nástroje pro automatické testování a ověřování vlastností implementovaných programů.

V současném stavu je provedena analýza a návrh základní struktury softwarového prostředí dle standardu IEEE1451.1. Tento standard definuje nezbytné třídy a komponenty prostředí pomocí textového popisu s rysy objektově-orientovaného specifikačního jazyka, doplněného diagramy chování nazvanými state machine. Jejich sémantika je blizká statchart diagramům jazyka UML, ve kterém je proveden kompletní návrh aplikačního prostředí v tomto projektu. To znamená, že převod IEEE1451 prostředí z reprezentace standardu do diagramů jazyka UML je velice přimočarý. Nicméně, složitost realizace je způsobena rozsahem prostředí standardem definovaném. Požadavky na realizaci celého standardu není možné v rámci této práce naplnit, a je tedy nutné zaměřit se na realizace pouze takové části, která bude samostaně použitelná a umožní pozdější rozšíření bez nutnosti významných a náročných změn. Toto je diskutováno v kapitole 3 dokumentu. Základní třídy lze rozdělit do tří skupin dle jejich významu pro aplikační prostředí. V každé z těchto skupin bude realizace omezena na implementaci vybraných tříd, například ve skupině tříd IEEE1451_Component, budou implmentovány podtřídy IEEE1451_Action, IEEE1451_Parameter a třída IEEE1451_ParameterWithUpdate a třídy od ní zděděné. Toto umožní realizovat funkční podmnožinu standardu a implementovat demonstrační aplikaci a základní množinu testů.

V textové části diplomové práce bude doplněn detailní popis a analýza realizované podmnožiny standardu, včetně diskuze obhajující zvolený postup. Dále bude práce obsahovat nezbytné informace pro potencionální řešitele projektu, jenž by stávající implementaci rozšířil pro ostatní části definovaná ve standardu. Samotná realizace bude

provedena v jazyce C# ve vývojovém prostředí Microsoft Visual Studio 2005. Kompletní návrh prostředí bude proveden v jazyce UML, za použití nástroje Microsoft Visio 2003. Cílem je při zachování běžných postupů při vývojí softwaru vytvořit přehlednou implementaci, která by zahrnovala kompletní programovou dokumentaci, umožňující snadné pochopení aplikačními programátory a možné budoucí rozšíření dle standardu.

Literatura

[1] IEEE Std 1451.1-1999, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Network Capable Application Processor (NCAP) Information Model, 1999.

[2] Handbook of sensor networks; algorithms and architectures, Edited by Ivan Stojmenovic, Wiley-Interscience, 2005.

[3] Distributed Sensor Networks", S. S. Iyengar, R. R. Brooks, Chapman & Hall/CRC, 2004.

[4] Programming C#, Jesse Liberty, O'Reilly, 2005.

[5] UML Pocket Reference, Dan Pilone, O'Reilly, 2003.