**TECHNICKÁ UNIVERZITA V LIBERCI**
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M 2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

# Internetová podpora výuky z prostředí Matlab a .NET

# Internet Oriented Support of Teaching with Matlab and .NET

**Diplomová práce**

Autor: **Daniel Reis**
Vedoucí diplomové práce: Doc. Ing. Osvald Modrlák, CSc.
Konzultant: Ing. Lukáš Hubka

**Liberec 15. 5. 2008**

**Prohlášení**

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že s o u **h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

**Poděkování**

Na tomto místě bych rád poděkoval vedoucímu diplomové práce panu Doc. Ing. Osvaldu Modrlákovi, CSc. a konzultantovi Ing. Lukáši Hubkovi za podporu při vypracování této diplomové práce.

Rád bych také poděkoval Pavlu Kratochvílovi za kritiku a rady týkající se tvorby internetových stránek.

# Abstrakt

Cílem této diplomové práce je zdokumentování možností využití funkcí aplikace Matlab v jiných programech, zvláště pak prostřednictvím sítě Internet. Zvláštní pozornost byla věnována nejnovějším technologiím v oblasti komunikace s aplikací Matlab a na poli tvorby interaktivních internetových stránek. V první částí práce je přehled možností komunikace s aplikací Matlab z jiného programu. Tato část obsahuje také detailní popis nejnovější technologie pro export funkcí aplikace Matlab kterou je Matlab Builder for .NET.

Druhá část je věnována praktické realizaci propojení aplikace Matlab s internetovou stránkou – modernizovanými stránkami podpory výuky. Na těch je názorně ukázán postup realizace internetových stránek využívajících funkcí aplikace Matlab. Výsledné stránky zároveň slouží jako vzor pro budoucí aplikace. Zmíněny jsou také problémy vzdáleného měření a řízení.

Klíčová slova: Matlab, Matlab Builder for .NET, .NET Framework, AJAX, ASP.NET

# Abstract

The aim of this diploma thesis is documentation of possibilities of inter-application communication with the Matlab, with special focus on the Internet communication. Extra attention was paid to the latest technologies of communication with the Matlab application and to interactive Internet pages technologies. The first part of the thesis contains a list of communication possibilities with Matlab application. A detailed description of Matlab Builder for .NET is also presented there.

The second part is dedicated to practical realization of interconnection between the Matlab and Internet pages – modernized pages for support of teaching of automated control. These pages are used for demonstration of creation process of Internet pages utilizing functions of the Matlab. Completed pages also serve as model for future similar applications. Problems of remote control and measurement are also mentioned.

Keywords: Matlab, Matlab Builder for .NET, .NET Framework, AJAX, ASP.NET

# Obsah

# Seznam ilustrací

**Seznam použitých termínů a zkratek**

AJAX zkratka anglického Asynchronous Javascript and XML, soubor technik pro tvorbu dynamických internetových stránek, umožňuje mj. komunikaci se serverem bez obnovení celé stránky

ASP zkratka anglického Active Server Pages, skriptovací jazyk pro HTTP servery vyvinutý firmou Microsoft

C/C++ programovací jazyk C nebo jeho rozšíření, programovací jazyk C++

C# programovací jazyk C# (čti „sí šárp“) vyvinutý firmou Microsoft; syntaxe vychází z jazyka C/C++, ale jedná se o moderní vysokoúrovňový programovací jazyk

CD zkratka anglického Compact Disc, optické datové medium

CGI zkratka anglického Common Gateway Interface, protokol pro komunikaci mezi aplikacemi

CIL zkratka anglického Common Intermediate Language - společný střední jazyk, programovací jazyk, do něhož jsou překládány zdrojové kódy ostatních programovacích jazyků prostředí .NET Framework

CLR zkratka anglického Common Language Runtime – společné prostředí běhu, součást Microsoft .NET Framework

CLS zkratka anglického Common Language Specification, specifikace prostředí od společnosti Microsoft, které umožňuje vyšším programovacím jazykům pracovat nezávisle na platformě; základ technologie .NET (ECMA-335 a ISO/IEC 23271)

CLSID zkratka anglického Class Identifier, jednoznačný identifikátor třídy, podtyp GUID

COM zkratka anglického Component Object Model, standard pro rozhraní a implementaci v objektů objektově orientovaném programování

CSS zkratka anglického Cascading Style Sheets – kaskádové styly, jazyk a technologie popisující vzhled prvků v markup dokumentu

| | |
|---|---|
| CTS | zkratka anglického Common Type System – společný systém typů, soubor všech datových typů využitelných v prostředí Microsoft .NET Framework |
| DCOM | zkratka anglického Distributed Component Object Model, rozšíření standardu COM |
| DDE | zkratka anglického Dynamic Data Exchange, standard pro komunikaci mezi aplikacemi |
| dll | zkratka anglického Dynamic Link Library, knihovna systému Microsoft Windows, OS/2; taktéž přípona souboru knihovny v OS Windows |
| DLL hell | anglický termín používaný souhrnně pro problémy nastávající při využití knihoven dll. Zvláště pak pro problémy nekompatibilních verzí knihoven, hromadění nepoužívaných knihoven či nemožnost vyhledat právě potřebnou knihovnu |
| DOM | zkratka anglického Document Object Model, standard popisující objektovou reprezentaci elementů HTML či XML stránek |
| GAC | zkratka anglického Global Assembly Cache, součást Microsoft .NET Framework, globální úložiště assembly (v rámci jednoho počítače) |
| GUID | zkratka anglického Globally Unique Identifier – globálně unikátní identifikátor, unikátní jméno komponentu ve standardu COM |
| GUI | zkratka anglického Graphical User Interface – grafické uživatelské rozhraní |
| HTML | zkratka anglického HyperText Markup Language, jazyk pro tvorbu Internetových stránek |
| HTTP | zkratka anglického HyperText Transfer Protocol, komunikační protokol pro přenos informací v prostředí Internetu, zvláště WWW |
| hypertext | obvykle textový dokument, obsahující odkazy na další dokumenty případně další funkce; hlavní část Internetových stránek |
| IIS | zkratka anglického Internet Information Services, soubor služeb a serverů firmy Microsoft pro obsluhu Internetových protokolů, součást některých operačních systému Windows |
| Internet | celosvětová informační síť propojená pomocí protokolu IP |

| | |
|---|---|
| JavaScript | interpretovaný skriptovací jazyk pro provádění skriptů na straně uživatele, v internetovém prohlížeči |
| JIT | zkratka anglického Just-in-Time, obvykle ve spojení JIT compilation / JIT kompilace - kompilace až před vykonáním kódu |
| JRE | zkratka anglického Java Runtime Environment, prostředí pro běh programů napsaných v jazyce Java |
| m-file | soubor zdrojového kódu v jazyce Matlab; přípona .m |
| mcc | zkratka anglického Matlab C compiler – kompilátor jazyka C aplikace Matlab, taktéž název příkazu aplikace Matlab |
| MCR | zkratka anglického Matlab Compiler Runtime, sada knihoven tvořící prostředí pro běh programového kódu zkompilovaného pomocí Matlab Compiler |
| PHP | rekurzivní zkratka z anglického PHP: Hypertext Preprocessor, open source skriptovací jazyk pro HTTP servery |
| RSS | v kontextu této práce Ústav řízení systémů a spolehlivosti Fakulty mechatroniky a mezioborových inženýrských studií Technické univerzity v Liberci |
| pointer | proměnná obsahující odkaz (adresu paměti) jiné proměnné, nikoli data |
| SDK | zkratka anglického Software Development Kit, soubor nástrojů a dokumentů, usnadňující programátorovi vývoj aplikace pro specifickou platformu či systém apod. |
| tag | klíčové slovo určující strukturu markup dokumentů |
| TCP/IP | zkratka anglického Transmission Control Protocol a Internet Protocol, protokol pro přenos dat v síti Internet a intranet |
| toolbox | v kontextu této diplomové práce se toolboxem rozumí rozšíření základní aplikace Matlab; například Control Toolbox přidává funkce pro řešení úloh automatického řízení |

| | |
|---|---|
| varargin | zkratka anglického Variable Arguments Input; klíčové slovo používané ve zdrojových kódech aplikace Matlab pro nespecifikovaný počet vstupních argumentů funkce |
| varargout | zkratka anglického Variable Arguments Output, viz varargin |
| W3C.org | zkratka anglického World Wide Web Consortium, hlavní organizace provádějící standardizaci WWW |
| WWW | zkratka anglického World Wide Web, systém hypertextových dokumentů přístupný pomocí sítě Internet |
| XML | zkratka anglického Extensible Markup Language, je specifikace pro tvorbu markup jazyků jako je například HTML |

# Úvod

S pokračujícím rozvojem sítě Internet a internetových stránek se stupňují požadavky na funkce které nabízejí. Tento prudký rozvoj s sebou přináší stále nové technologie v jejichž záplavě může být obtížné se orientovat. Důvodem vzniku této diplomové práce byla potřeba zdokumentovat poslední vývoj možností spolupráce aplikace Matlab s Internetem. Zároveň má být tato práce snadno použitelným návodem na propojení aplikace Matlab s Internetem pomocí nejnovějších technologií, konkrétně Matlab Builder for .NET.

V současné době je v Ústavu řízení systémů a spolehlivosti (RSS) jako hlavní prostředek spojení aplikace Matlab s Internetem používán Matlab Web Server v kombinaci se skriptovacím jazykem PHP a HTTP serverem Apache. Protože Matlab Web Server přestal být podporován a vyvíjen, objevila se nutnost prozkoumat současnou situaci a nalézt alternativní cesty, jak spojit aplikaci Matlab s Internetem.

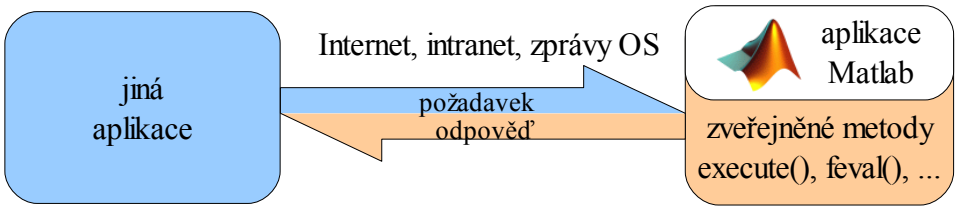
Cílem této diplomové práce se tedy stalo prozkoumat perspektivní možnosti komunikace aplikace Matlab s ostatními programy a zhodnotit jejich možné výhody a omezení při různých způsobech využití, konkrétně na stránkách podpory výuky předmětů automatického řízení a v e-laboratoři spojitého řízení. Zároveň bylo třeba inovovat stránky podpory výuky předmětů automatického řízení. To umožnilo zdokumentovat použití nejnovějšího prostředku pro export funkcí z aplikace Matlab, Matlab Builder for .NET, který je pro takovou aplikaci velmi vhodný. Zároveň tak bylo možné demonstrovat výhody využití moderních technologií při tvorbě interaktivních internetových stránek. Navrhované stránky mají umožnit uživateli vytvořit několik modelů dynamických systémů. Na takto vytvořených systémech pak má být umožněno provádět různé operace demonstrující možnosti aplikace Matlab při analýze dynamických systémů. Hlavní podmínkou bylo řešení nevyžadující instalaci žádných programů na straně koncového uživatele a přiměřená rychlost i na pomalejším připojení.

Posledním cílem bylo vytvořit podmínky pro vzdálené měření v e-laboratoři spojitého řízení s využitím nejnovějších technologií komunikace aplikace Matlab.

Jako prostředky realizace těchto cílů byly zvoleny technologie JavaScript, CSS a DOM pro vytvoření interaktivních stránek na straně uživatele. Prostředí .NET Framework a Matlab Builder for .NET pro integraci funkcí aplikace Matlab s HTTP serverem a techniky AJAX pro dynamickou komunikaci mezi uživatelem a serverem.

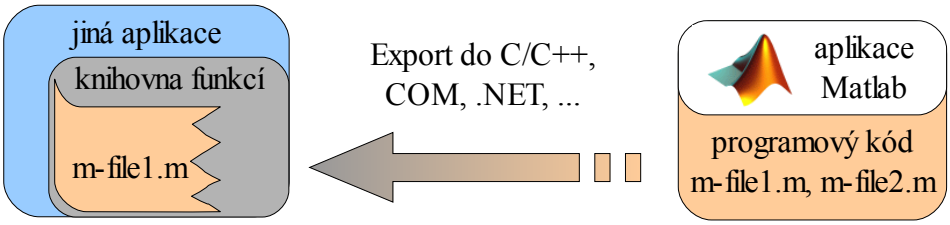
# 1 Prostředky komunikace Matlab 2006

Pokud komunikaci definujeme jako možnost využít funkcí aplikace Matlab z jiného programu, pak umožňuje aplikace Matlab dva principiálně odlišné způsoby, jak toho dosáhnout. První možností, jak zpřístupnit funkce a programy aplikace Matlab po síti a Internetu je přistoupit přímo k samotné aplikaci. Pak je na počítači fungujícím jako server spuštěna samostatná aplikace Matlab, obdobně jako by ji spustil uživatel sám. Ta dále vykonává požadované úlohy.




*Ilustrace 1: Přímá komunikace s aplikací Matlab*

Druhou možností zpřístupnění funkcí aplikace Matlab, která se rozvíjí zvláště poslední dobou, je využití možnosti automatického překladu programu v jazyce Matlab do jazyka jiného. Takto přeložený kód je pak možné zkompilovat do podoby komponentu některého z podporovaných standardů. Komponent je následně využitelný v libovolné aplikaci, a pokud je touto aplikací například HTTP server, můžeme funkce aplikace Matlab zpřístupnit po Internetu.




*Ilustrace 2: Export funkce z aplikace Matlab*

Obě tyto cesty zpřístupnění kódu je možno realizovat několika způsoby. Jejich stručný výčet je uveden v následujících kapitolách.

# 1.1 Vzdálený přístup k aplikaci Matlab

## 1.1.1 Component Object Model COM, DCOM

Standard COM, představený společností Microsoft v roce 1993, popisuje předpokládané chování objektů nezávisle na platformě či programovacím jazyku – mimo jiné definuje, jakým způsobem spolu budou komunikovat jednotlivé objekty (komponenty) v objektově orientovaném programování [1]. Vzhledem k tomu, že i aplikace jako celek je objekt, spadá do této definice i komunikace mezi aplikacemi. Objekt – aplikace může zpřístupnit některé své metody a tím umožnit ostatním objektům – programům interakci.

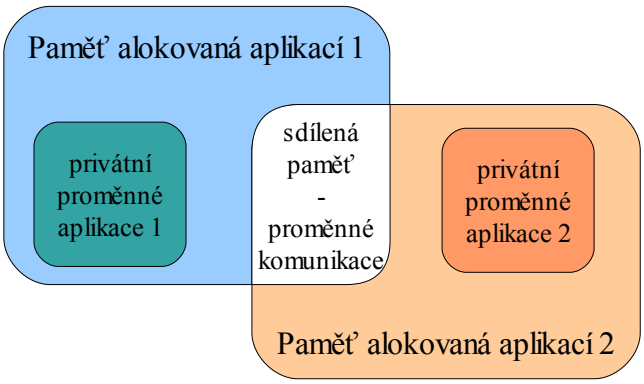
Standard určuje konkrétní způsoby realizace charakteristik objektově orientovaného programování, jako je dědění, polymorfismus, zapouzdření, reflexe atd. Mimo jiné definuje, že každý komponent musí být schopen zodpovědět dotaz na svá rozhraní (metoda QueryInterface). Tato skutečnost umožňuje práci s komponentem, který jsme nevytvořili a o kterém nic nevíme. Jediné co potřebujeme znát je jméno komponentu. Každý komponent musí mít své globálně unikátní jméno, tzv. globally unique identifier (GUID), a před použitím musí být zaregistrován v registrech operačního systému. Tato skutečnost s sebou přináší i úskalí v podobě konfliktů různých verzí komponentů (stejné jméno, ale jiné GUID), hromadění komponentů které již nejsou využívány, ale stále jsou evidovány v registrech, apod.

Konkrétně k aplikaci Matlab můžeme přistoupit pomocí jejího COM rozhraní jako ke třídě „Matlab Application class“. Při pokusu o vytvoření instance této třídy se program podívá do registrů operačního systému, kde vyhledá na adrese HKEY_CLASSES_ROOT\Matlab.Application\CLSID identifikátor třídy kterou se snažíme vytvořit. CLSID je zkratka anglického Class Identifier. Je to podskupina GUID sloužící k identifikaci tříd. Během vytvoření instance se spustí samotná aplikace Matlab. Voláním metod třídy „Matlab Application class“ bude náš program interagovat přímo s takto spuštěnou aplikací Matlab. Například metoda *execute()* pak umožňuje vykonat příkaz stejně, jako bychom ho zadali do příkazové řádky ručně spuštěné aplikace.

Rozšíření standardu COM o síťovou komunikaci se nazývá Distributed Component Object Model (DCOM). Umožňuje přistupovat ke komponentům nejen v rámci jednoho počítače, ale také po síti, včetně sítě typu TCP/IP.

### 1.1.2 Přímý přístup do paměti

Jednou z možností, jak přistoupit k proměnným aplikace Matlab, je využít sdílené paměti. Na Internetu je možné nalézt několik článků na toto téma, včetně hotových knihoven [2]. Komunikace přes sdílenou paměť je velmi efektivní a rychlá, omezuje se však na předávání dat. Příkazy musíme aplikaci Matlab předávat jiným způsobem, například pomocí rozhraní COM. Toto řešení přináší úskalí v podobě náročnosti implementace na znalosti programátora. Nesprávné zapsání do paměti jiného programu způsobí pravděpodobně havárii nebo poškození dat. Protože při tomto způsobu komunikace se data nikam nepřenášejí ani nekopírují, je vhodný spíše pro použití v případech kdy je kritická rychlost přenosu dat mezi aplikací Matlab a klientským programem. Nebo v případě, že přenášíme velký objem dat. Již z principu je tento způsob možné použít pouze při komunikaci mezi aplikacemi na jednom počítači. Pro komunikaci po síti je nutné napsat program tvořící mezičlánek mezi aplikací Matlab a sítí. Protože však přenosová rychlost počítačové sítě bude s velkou pravděpodobností nižší než rychlost přenosu dat pomocí některého s vyšších rozhraní pro komunikaci mezi aplikacemi, toto řešení se pro většinu úloh jeví jako příliš komplikované.



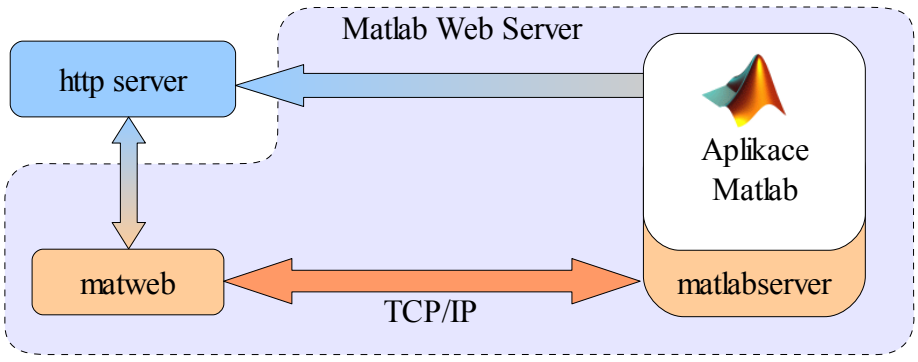

*Ilustrace 3: Princip komunikace pomocí sdílené paměti*

### 1.1.3 Common Gateway Interface

Common Gateway Interface (CGI) je protokol pro volání externího kódu z HTTP serveru. Umožňuje serveru obrátit se na jiný spustitelný kód, ať již se jedná o skript či samostatnou aplikaci, předat mu požadavek a přijmout odpověď. CGI vznikl před rozšířením skriptovacích jazyků pro HTTP servery jako způsob, jak vygenerovat na serveru dynamický obsah. Pomocí CGI se z HTTP serveru můžeme obrátit přímo na aplikaci Matlab, nebo na kód exportovaný pomocí Matlab Compiler do podoby spustitelného programu [15]. Zároveň je CGI součástí Matlab Web Server.

### 1.1.4 Matlab Web Server

Hlavním nástrojem na propojení aplikace Matlab s Internetem byl v jejích předchozích verzích Matlab Web Server. Na rozdíl od standardů komunikace mezi aplikacemi, zmíněných v předchozích kapitolách, se nejedná o technologii či standard, ale o specializované programové rozšíření aplikace Matlab. Matlab Web Server umožňuje oboustrannou komunikaci mezi aplikací Matlab, HTTP serverem a uživatelem pomocí CGI rozhraní a formulářů jazyka HTML. Matlab Web Server se skládá ze dvou hlavních částí. *Matlabserver* je TCP/IP server zajišťující komunikaci mezi aplikací Matlab a HTTP servery. Běží na počítači kde je nainstalována aplikace Matlab. *Matweb* je klient matlabserver. Přijímá požadavky HTTP serveru pomocí rozhraní CGI a odesílá je pomocí TCP/IP na matlabserver. Grafický výstup je potřeba na HTTP server přenést zvlášť. Podrobnější informace lze získat například v [16], [17].




*Ilustrace 4: Schéma fungování Matlab Web Server*

## 1.2 Vzdálený přístup k programovému kódu Matlab

Všechny varianty exportu funkčního kódu z aplikace Matlab používají v jedné fázi procesu kompilace Matlab Compiler. Pomocí něj je možno zkompilovat většinu funkcí aplikace Matlab, existují však jistá omezení [22], [23], se kterými je nutné počítat.

### 1.2.1 Matlab Compiler

Součástí aplikace Matlab je i kompilátor Matlab Compiler (mcc). Ten umožňuje převést kód v jazyce Matlab (m-file) do jazyka C/C++ a vytvořit tak samostatně spustitelný program nebo komponent použitelný při tvorbě další aplikace. Takový program pak můžeme volat například z libovolného skriptovacího jazyka pro HTTP servery (PHP, ASP, atd.) nebo pomocí CGI. Nadstavbou nad Matlab Compiler je řada nástrojů Matlab Builder. V kombinaci s touto nadstavbou umožňuje Matlab Compiler vytvářet širokou paletu komponentů různých standardů a využití.

Hlavním faktorem, který omezuje možnosti nasazení Matlab Compiler je neúplná podpora funkcí aplikace Matlab. Existují funkce a části aplikace Matlab, které není možné exportovat pomocí Matlab Builder. Mezi nepodporovanými funkcemi jsou některé méně důležité, jako grafická uživatelská rozhraní (GUI), avšak také některé velmi užitečné a často používané nástroje aplikace Matlab, jako například Matlab Simulink [22], [23].

### 1.2.2 Matlab Builder

Novinkou nahrazující od verze Matlab 2006a Matlab Web Server je Matlab Builder [3]. Jedná se o nadstavbu pro Matlab Compiler schopnou vytvářet samostatné komponenty různých standardů, kreré obsahují funkce exportované z aplikace Matlab. Existují tři varianty Matlab Builder, lišící se aplikací či standardem, pro který vytváří komponenty.

*Matlab builder for Excel* vytváří ze zdrojového kódu aplikace Matlab modul plug-in pro aplikaci Microsoft Excel. *Matlab builder for Java* vytváří třídy jazyka Java a *Matlab builder for .NET* umožňuje vytvořit knihovnu standardu COM nebo .NET.

Vzhledem k tomu, že se jedná o nejnovější technologii, která zřejmě určuje budoucí

vývoj na poli zpřístupňování funkcí aplikace Matlab ostatním programům, je fungování Matlab Builder for .NET věnována samostatná kapitola. Protože všechny varianty Matlab Builder využívají při kompilaci Matlab Compiler, platí pro ně i stejná omezení využitelných funkcí [24].

## 1.3 Výběr prostředku komunikace

Při volbě prostředku propojení aplikace Matlab s jiným programem určuje naše rozhodnutí několik faktorů. Aplikace Matlab existuje ve verzi pro operační systémy Windows i Linux. Prvním omezením je, že některé komunikační prostředky jsou specifické pro prostředí daného operačního systému. Například standardy COM a .NET jsou omezeny na platformu Windows. Dalším omezením je u exportu programového kódu rozsah použitelných funkcí – Matlab Compiler a potažmo Matlab Builder podporuje většinu, nikoli však všechny funkce aplikace Matlab [22], [23], [24]. Vážným omezením může být například absence podpory Matlab Simulink [22]. Naproti tomu volání samotné aplikace Matlab nemá téměř žádná omezení oproti aplikaci spuštěné uživatelem ručně. Je v podstatě zaručeno, že program bude fungovat stejně, jako by ho spustil uživatel sám. Mezi aplikací Matlab obsluhovanou myší a klávesnicí a aplikací obsluhovanou například pomocí rozhraní COM není žádný funkční rozdíl. Pouze některé grafické výstupy, jako například grafická uživatelská rozhraní, je potřeba nahradit či upravit.

Pokud se rozhodneme pro export kódu, musíme zvážit v jakém prostředí chceme výsledný komponent použít. Všechny varianty Matlab Builder jsou samostatně prodejné [18] a pokrývají pouze specifickou skupinu cílových programů. Samostatně je možné zakoupit také základní Matlab Compiler [18] vytvářející komponenty pro C/C++ a samostatně spustitelné programy.

# 2 Matlab Builder for .NET

Matlab Builder for .NET je rozšířením Matlab Compiler, které umožňuje využít programový kód Matlabu v prostředí jazyků CLS (Common Language Specification). Jedná se o poslední technologii pro zakomponování funkcí aplikace Matlab do jiného programu.

## 2.1 Microsoft .NET Framework

Základ, na kterém je vystavěn Matlab Builder for .NET, tvoří Microsoft .NET Framework. Je to rozsáhlé prostředí pro vývoj a běh programů, jehož cílem je usnadnit, zrychlit a zefektivnit vývoj nových aplikací. Tohoto účelu má dosáhnout několika základními vlastnostmi: *součinnost* - .NET Framework obsahuje prostředky pro komunikaci mezi aplikacemi a to i pomocí starších technologií, jako je například COM. *Společné prostředí běhu* – programový kód napsaných v některém z programovacích jazyků .NET je překládán do jednotného programového jazyka, tzv. Common Intermediate Language (CIL). V tomto jazyce je program distribuován a až před samotným spuštěním je provedena kompilace, tzv. just-in-time compilation (JIT). Kombinace těchto dvou konceptů (CIL a JIT) se nazývá Common Language Runtime (CLR), tedy společné prostředí běhu. *Přenositelnost* – protože jazyk CIL je nezávislý na platformě, je možné program vytvořený v prostředí .NET Framework nasadit kdekoli, kde je možné nainstalovat framework samotný jakožto jedinou podmínku běhu. *Jazyková nezávislost* - .NET Framework obsahuje soubor všech přípustných datových typů a jejich interakcí, které je možno v tomto prostředí používat. Tento soubor se nazývá Common Type System (CTS). To umožňuje výměnu proměnných mezi programy, napsanými v různých programovacích jazycích, které splňují standardy .NET Framework. *Zjednodušené nasazení* – součástí specifikací .NET Framework je pokus o zjednodušení procesu instalace a nasazení programů v něm vytvořených. Reaguje tak na časté problémy vznikající v předchozích standardech, například DLL Hell.

## 2.2 Princip funkce Matlab Builder for .NET

Základním výstupem Matlab Builder for .NET je knihovna dll. Matlab Builder for .NET umožňuje vytvářet knihovny standardu COM či .NET, rozdíl je pouze ve struktuře výsledné knihovny. Pro vytvoření knihovny standardu .NET je nutné mít nainstalovaný .NET Framework. Každá knihovna obsahuje jednu nebo více tříd. Tyto třídy jsou definovány uživatelem během procesu tvorby knihovny a může jich být vytvořen libovolný počet. Každá třída má jednu či více metod, které představují funkce exportované z programu Matlab. Jména tříd a metod obsažených ve výsledné knihovně musí být unikátní v rámci jedné knihovny, která zároveň tvoří prostor jmen (namespace) stejného jména jako má sama. V průběhu návrhu si uživatel přidáním jednotlivých funkcí v podobě souborů m-file zvolí, jaké metody má třída obsahovat.

Během samotné kompilace vytvoří builder potřebnou strukturu knihovny, zařídí implementaci a převody datových typů. Dále zajistí předávání výjimek, které nastanou při vykonání kódu jazyka Matlab, nadřazenému programu. Dalším krokem je kontrola závislostí. Pokud v exportovaném kódu voláme jinou funkci, bude tato automaticky přidána do knihovny tak, aby výsledný program fungoval jako v prostředí Matlab. Takto automaticky přidané funkce nejsou dostupné uživateli, slouží pouze pro vnitřní potřebu knihovny a vně knihovny nejsou vidět. Na závěr se provede překlad zdrojového kódu a sestavení samotné knihovny pomocí Matlab Compiler. Výsledná knihovna sestává ze dvou souborů, jeden s příponou .dll a druhý s příponou .ctf. Pro běžné funkce se velikost tohoto souboru pohybuje kolem 3 MB a můžeme jí ovlivnit omezením počtu prohledávaných toolboxů v nastavení projektu (záložka „Toolboxes on Path“).

## 2.3 Matlab Compiler Runtime

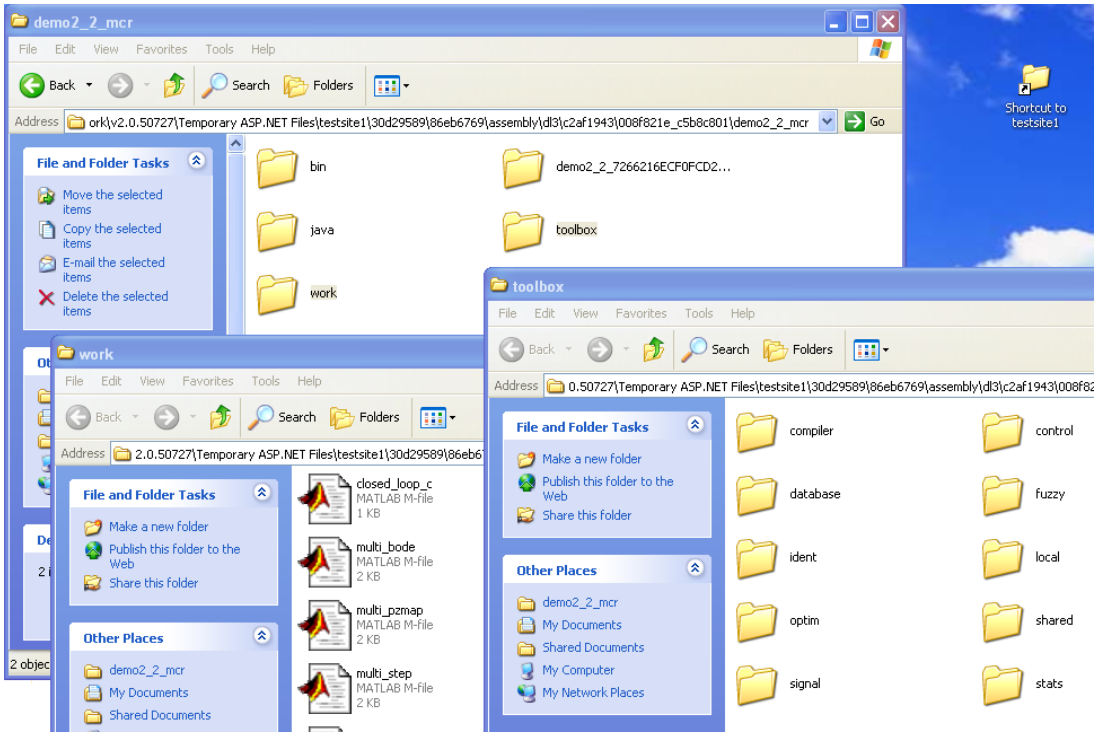
K běhu jakékoli aplikace zkompilované pomocí Matlab Compiler, tedy i komponentu vytvořeného pomocí Matlab Builder, je potřeba tzv. Matlab Component Runtime (MCR). Jedná se o sadu knihoven, které vytvářejí na cílovém počítači prostředí pro běh zkompilovaného programu [4].

Podle adresářové struktury a souborů, které vytváří MCR za běhu uživatelského programu a která nápadně připomíná strukturu adresáře samotné aplikace Matlab (viz

ilustrace 5) a také podle náznaku principu vnitřního fungování tříd vytvořených kompilátorem (viz 2.3.1), se zdá, že hlavní částí je speciálně upravené jádro aplikace Matlab.

Pro programátora využívajícího funkce exportované pomocí Matlab Builder for .NET je nejdůležitější součástí knihovna MWArray.dll, s kterou musí při návrhu svého programu přímo pracovat. Tato knihovna je podrobně popsána v kapitole 2.3.1.

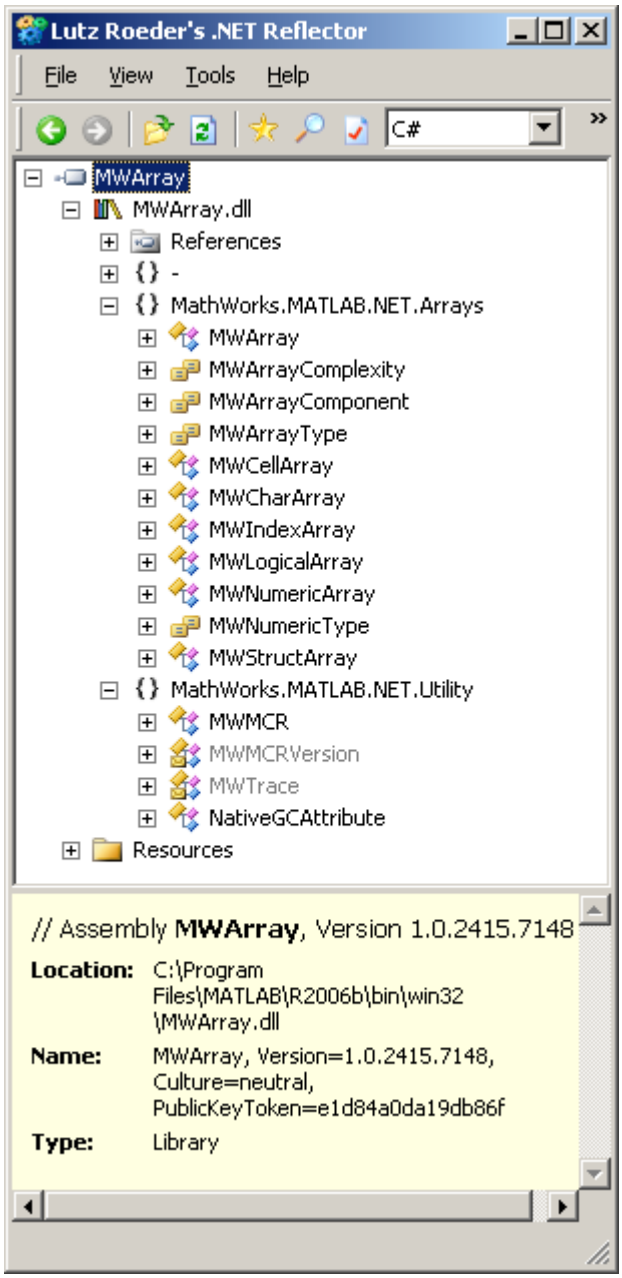
Matlab Compiler Runtime je závislá na verzi aplikace Matlab, ve které jsme provedli sestavení knihovny. Na cílovém počítači musí být vždy nainstalována verze MCR odpovídající verzi zdrojové aplikace Matlab [4].




*Ilustrace 5: Adresářová struktura vytvořená MCR při spuštění kódu z knihovny*

### 2.3.1 Struktura knihovny MWArray.dll

Pro vytvoření programu, který využívá komponent vytvořený pomocí Matlab Builder for .NET, je nezbytné porozumět struktuře knihovny MWArray.dll. Ta obsahuje definice všech datových typů, které používají komponenty exportované z aplikace Matlab.

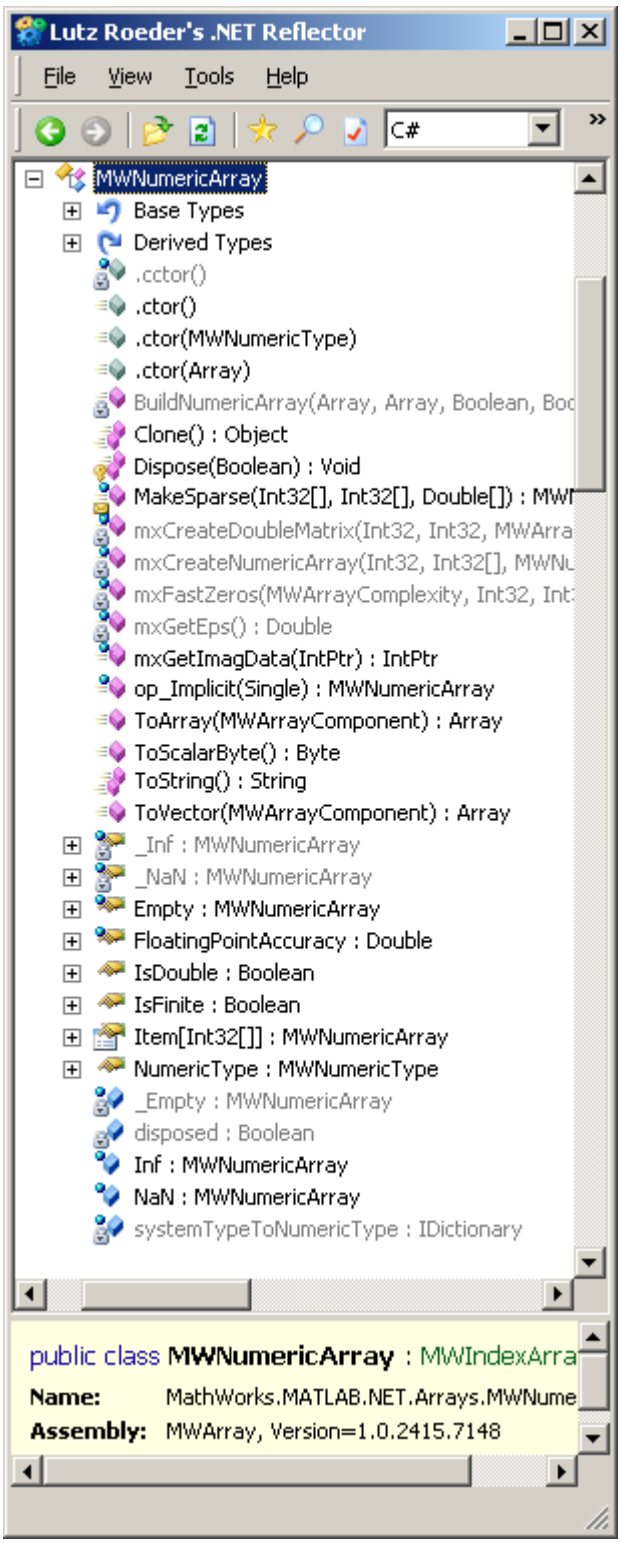



*Ilustrace 6: Struktura knihovny MWArray*

Strom začíná seznamem referencí, za nímž následuje název prostoru jmen (označen { }). Knihovna obsahuje dva prostory jmen: *Arrays* a *Utility*. Vidíme, že prostor jmen Arrays obsahuje třídy odpovídající jednotlivým typům matic, které MCR rozeznává. Od abstraktního typu MWArray jsou odvozeny typy další, jako například numerická matice, matice znaků apod.

Druhý prostor jmen, Utility, obsahuje třídu MWMCR umožňující propojení se samotným výpočetním jádrem MCR. Instance tříd vytvořených pomocí Matlab Builder for .NET obsahují privátní pointer odkazující právě na instanci třídy MWMCR.

Ve spodní části ilustrace vidíme informace o zvoleném prvku, v tomto případě o celé assembly MWArray. Za zmínku stojí položka jméno (Name), která ji jednoznačně identifikuje (například pro vložení do externích referencí programu).




*Ilustrace 7: Struktura třídy MWNumericArray*

Pokud rozbalíme definici například třídy MWNumericArray (numerická matice), uvidíme že obsahuje deklaraci několikrát přetíženého konstruktoru (metoda .ctor()) a řadu privátních i veřejných metod (označeny fialovou ikonou). Dále obsahuje několik veřejných i privátních vlastností. Po kliknutí na jakoukoli položku ze seznamu se v dolním informačním okně zobrazí podrobné informace, například deklarace u metod.

K zobrazení struktury knihovny MWArray.dll byl použit program *Lutz Roeder's .NET Reflector*. Ten využívá reflexe (schopnosti programu pozorovat svou vlastní strukturu) k zobrazení vnitřní struktury komponenty typu .NET [5].

*Poznámka:* Pro názornost jsou z ilustrace odstraněny přibližně ¾ atributů třídy MWNumericArray.

## 2.4 Omezení zdrojového kódu pro export

Návratová hodnota a argumenty jakékoli funkce exportované pomocí Matlab Builder musí být jednoho z typů definovaných v knihovně MWArray.dll, která je součástí MCR. Konkrétně se jedná o typy odpovídající numerické, znakové a logické matici, navíc matici typu „cell array“ - matici obsahující jiné matice a „struct array“ - ekvivalent struktury jiných programovacích jazyků. Posledním definovaným typem je typ „index array“, což je podle neoficiální dokumentace [13] abstraktní třída sloužící k implementaci indexování matic v podobě jakou používá aplikace Matlab. Tomu poněkud odporuje skutečnost, že tato třída má definovaný veřejný konstruktor. Bližší informace se nepodařilo nalézt, nicméně matici tohoto typu je zřejmě také možno použít jako argument či návratovou hodnotu.

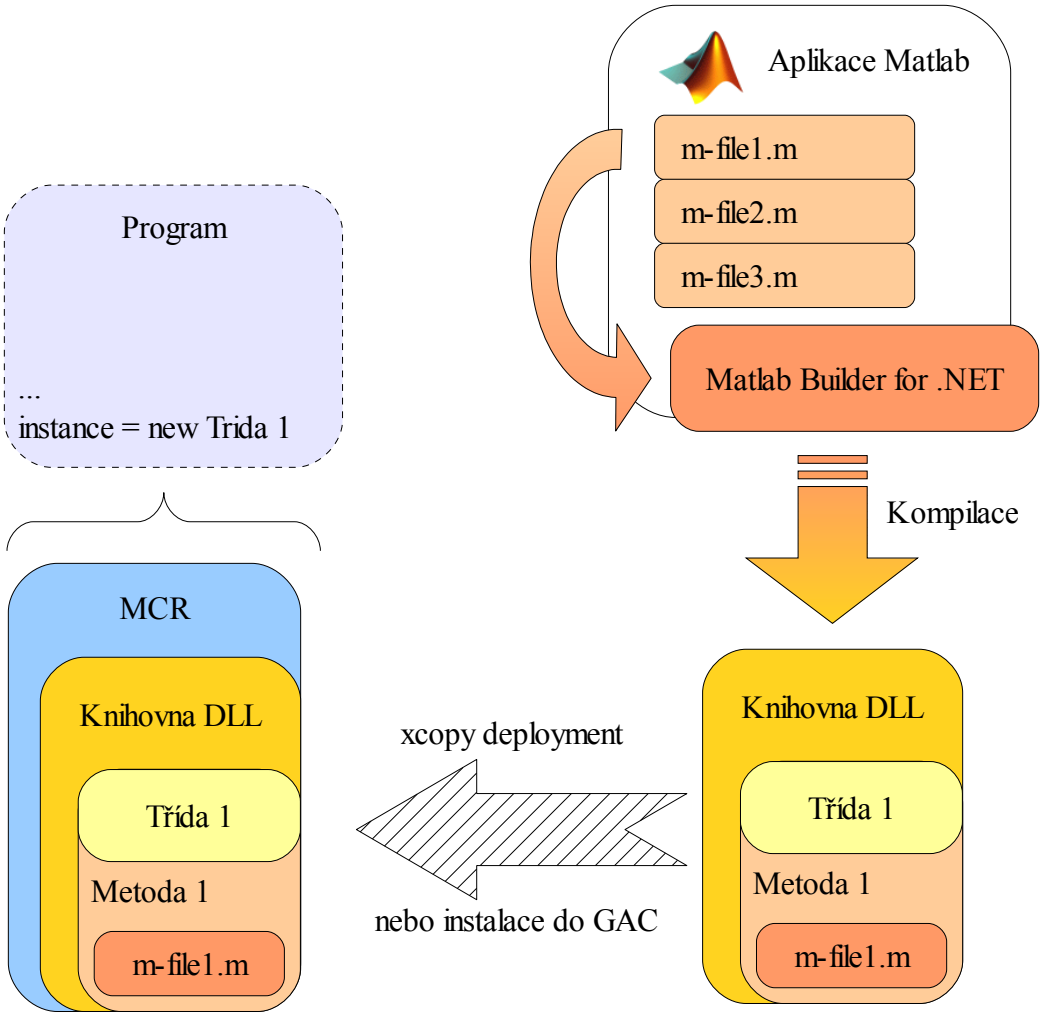
Konkrétním následkem tohoto omezení použitelných typů je, že exportovaná funkce nemůže například vracet jakýkoli obrazový přenos, i když v samotné aplikaci Matlab tato možnost existuje. Další nepříjemnou skutečností je, že Matlab Builder na vzniklý problém nijak neupozorní, knihovnu obsahující takovou funkci zkompiluje bez chyby. Výsledkem je ovšem metoda, která vrací vždy prázdný výsledek (NULL). Lze předpokládat, že důvodem tohoto opatření je zabránit algoritmizaci mimo exportovanou funkci. Takto musí všechny metody vracet konkrétní výsledky a nikoli jejich dílčí části. V případě, že chceme uživateli zpřístupnit proměnnou jiného typu než jsou výše vyjmenované, můžeme ji standardně uložit na disk pomocí funkce save, stejně jako v aplikaci Matlab. Toto řešení můžeme nouzově použít i pro uložení proměnných, které chceme předávat mezi jednotlivými funkcemi (metodami) a které nejsou jednoho z typů definovaných v knihovně MWArray.dll.

V případě že je potřeba, aby funkce měla i obrazový výstup, využijeme funkce aplikace Matlab *print()* [6]. Pomocí této funkce uložíme obsah kterékoli figury programu Matlab do souboru na disk, kde s ním můžeme dále libovolně pracovat.

Při návrhu aplikace je také nutné počítat s omezeními využitelných funkcí. I když Matlab Builder umožňuje exportovat z aplikace Matlab většinu jejích funkcí, platí určitá omezení [24]. Například není možné exportovat kód využívající Matlab Simulink.

## 2.5 Použití komponentu z Matlab Builder for .NET

Výsledným produktem kompilace pomocí Matlab Builder for .NET je knihovna, která obsahuje funkce zvolené pro export. Před jejím využitím si koncový uživatel musí nainstalovat MCR a odpovídající verzi .NET Framework. Matlab Builder for .NET je schopen pracovat s .NET Framework verze 1.1 a od verze aplikace Matlab 2006b i s verzí 2.0 [14]. Využití zkompilované knihovny pak již spočívá v pouhém vytvoření instance jedné z jejích tříd a využití metod této instance.



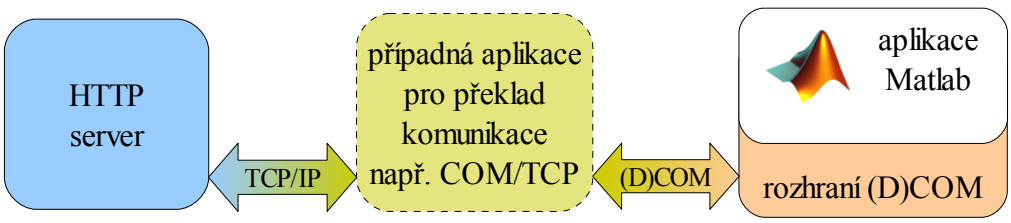

*Ilustrace 8: Schema principu funkce Matlab Builder for .NET*

# 3 Praktické propojení Matlab a Internet

Z kapitoly 1 vyplývá, že všechny možnosti propojení aplikace Matlab s libovolným dalším programem můžeme rozdělit na dvě kategorie: propojení se samotnou aplikací Matlab a nebo export programového kódu z aplikace Matlab a jeho následné začlenění do aplikace jiné. Protože přenesení výpočetních funkcí až na stranu uživatele není vhodné – vyžaduje instalaci různých podpůrných programů a podobně – praktická řešení budou využívat hlavně propojení HTTP serveru a aplikace Matlab. Pro úplnost je uvedena i nejschůdnější cesta, jak provést výpočet na straně uživatele.

## 3.1 Propojení s aplikací Matlab pomocí COM

První praktickou možností je propojit aplikaci Matlab s HTTP serverem pomocí rozhraní (D)COM. Toto řešení je také doporučeno výrobcem [15]. Umožní spustit a ovládat plnohodnotnou aplikaci Matlab, což usnadní ladění a není třeba psát speciální funkce pro export. Při přímém propojení s aplikací Matlab potřebujeme samozřejmě licenci, nicméně jedna instalace aplikace Matlab dokáže obsloužit více připojení pomocí (D)COM. Tato možnost se jeví jako obzvláště vhodná pro práci s reálnými úlohami – vzdálené měření a řízení. Na řídícím počítači takové úlohy je zpravidla aplikace Matlab již nainstalována, s nákupem licence tedy není problém. Tím, že se připojíme k samotné aplikaci Matlab, se vyhneme všem potenciálním problémům, které mohou nastat při překladu zdrojových kódů pro export. Pokud je při měření používán Matlab Simulink, není možné využít export programového kódu [23] a zbývá pouze interakce se samotnou aplikací Matlab. Pokud by se cesta ovládání pomocí DCOM ukázala jako obtížně průchodná, je možné napsat jednoduchou aplikaci zajišťující překlad komunikace mezi protokolem COM a libovolným jiným protokolem.




*Ilustrace 9: Propojení HTTP serveru s aplikací Matlab s využitím COM*

## 3.2 Kompilace kódu jako samostatné aplikace

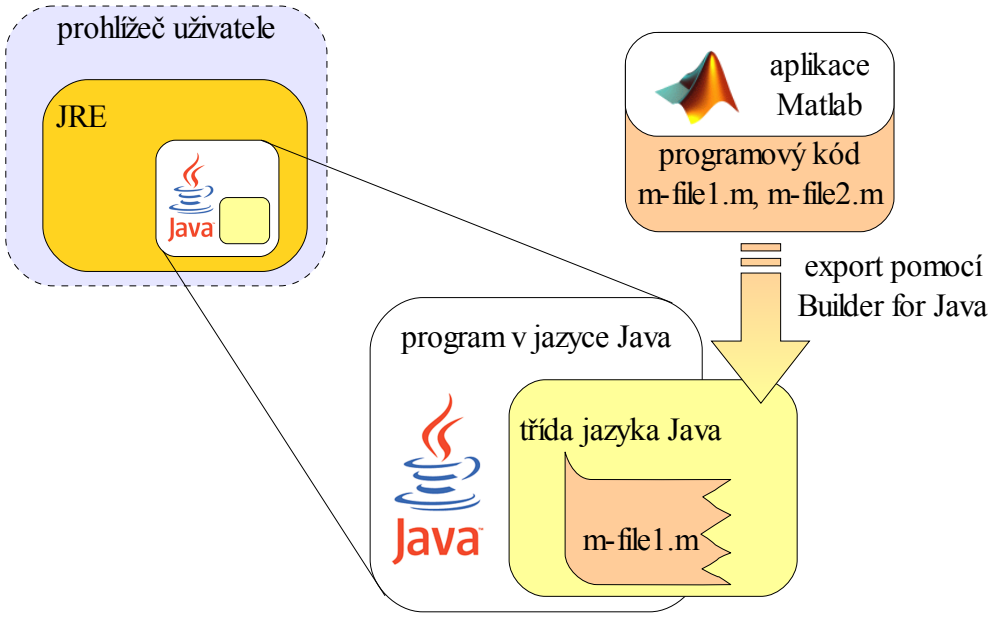
Základní možností, jak exportovat programový kód z aplikace Matlab, je překlad do podoby samostatně spustitelné aplikace pomocí Matlab Compiler. K tomu je potřeba zakoupit pouze licenci Matlab Compiler. S výslednou aplikací pak může HTTP server komunikovat například pomocí CGI. Výhodou tohoto řešení je snadné nasazení a možnost využití i na operačních systémech Unix/Linux. Pokud má nasazení probíhat v prostředí Unix/Linux, musí i kompilace proběhnout verzí Matlab Compiler určenou pro toto prostředí [29]. Je potřeba mít na paměti i omezení platná pro Matlab Compiler, viz kapitola 1.2.1 a [22], [23].

## 3.3 Export funkčního kódu pomocí Builder for .NET

Nejnovějším nástrojem pro zpřístupnění funkcí aplikace Matlab na Internetu je Matlab Builder for .NET. Jak bylo popsáno v kapitole 2.2, Builder for .NET umožňuje vytvořit komponent standardu COM či .NET. Tento komponent pak můžeme využít ve skriptovacím jazyce na HTTP serveru. Podrobným popisem jedné z variant tohoto postupu se zabývá kapitola 4. Použití Matlab Builder for .NET je vhodné zejména pro jednodušší aplikace, které je možno realizovat v několika jednoduchých funkcích. Export složitých funkcí využívajících několik toolboxů apod. může přinášet určité komplikace. U složitějších aplikací (například typu vzdáleného řízení) může být výhodnější použít některou z alternativ kvůli omezením funkcí použitelných v exportovaném kódu [24]. Viz diskuse v kapitole a 5.

## 3.4 Export funkčního kódu pomocí Builder for Java

Jediný praktický způsob, jak dostat funkční kód aplikace Matlab až do prohlížeče uživatele, je pomocí jazyka Java. Ten může běžet ve speciálním prostředí (Java Runtime Environment) téměř na libovolné platformě. Toto prostředí je zdarma ke stažení [21]. Podle statistik má více než 90 % uživatelů sítě Internet v současné době prohlížeč schopný spustit program v jazyce Java [25]. Nicméně pro uživatele stále platí nutnost nainstalovat správnou verzi MCR, čímž se snižuje komfort tohoto řešení. Pomocí Matlab Builder for Java provedeme export funkcí z aplikace Matlab, ty pak začleníme do programu v jazyce Java a odešleme uživateli. To může být výhodné například pokud by bylo kódem třeba zpracovat velké množství dat uložených v počítači uživatele. Kromě nutnosti instalovat MCR u uživatele jsou nevýhody obdobné jako u Matlab Builder for .NET, které byly zmíněny v kapitole 3.3. Před nasazením by bylo vhodné porovnat rychlost výpočtu pomocí funkce exportované do třídy jazyka Java a do knihovny standardu .NET. Jazyk Java nebyl primárně určen pro numerické výpočty a je možné, že bude pomalejší. Nicméně tato problematika je velmi složitá a zachází nad rámec této práce. Podrobnější informace je možno získat například v [26], [27], [28].




*Ilustrace 10: Export funkčního kódu pomocí Builder for Java*

# 4 Stránky podpory výuky automatického řízení

Pro modernizaci stránek podpory výuky automatického řízení, které doposud fungovaly na základě Matlab Web Serveru, jsem vybral Matlab Builder for .NET. Matlab Builder je prezentován jako přímý nástupce již nepodporovaného Matlab Web Serveru. Pro nenáročné výpočty, které se na těchto stránkách budou provádět, se jeví jako vhodný prostředek. Zároveň řešení této úlohy umožnilo zmapovat fungování Matlab Builderu a vytvořit jednoduchý návod pro jeho další využití v podobných aplikacích. Tento návod by měl být srozumitelný i pro ty, kteří nemají zkušenosti s využitím technologií .NET. Dokumentace Matlab Builder for .NET přímo na internetových stránkách výrobce www.mathworks.com je velmi podrobná a rozsáhlá. Přesto se při použití Matlab Builder for .NET objevilo několik problémů, které v dokumentaci nejsou postiženy a jejichž řešení zde předkládám.

## 4.1 Tvorba knihovny výpočetních funkcí

### 4.1.1 Návrh funkcí v prostředí Matlab

Prvním krokem je vytvoření samotné funkce, kterou budeme z aplikace Matlab exportovat. Při jejím návrhu musíme počítat s omezeními popsanými v kapitole 2.4. Chceme pracovat s modely dynamických systémů, nicméně tyto systémy nemůžeme použít v podobě, v jaké je používá aplikace Matlab. Pokud vytvoříme dynamický systém například funkcí tf(), je výsledkem proměnná typu „transfer function“ - přenos. Tento datový typ ovšem mimo aplikaci Matlab nemáme jak implementovat, knihovna MWArray.dll jej neobsahuje. Pokud tedy ve funkci chceme například vykreslit přechodové charakteristiky dvou uživatelem zadaných přenosů, musíme předat informace o těchto přenosech v datovém typu obsaženém v knihovně MWArray.dll. Pro obrazový přenos se nabízí předávat hodnoty jednotlivých koeficientů polynomu v čitateli a ve jmenovateli, protože můžeme realizovat proměnné typu matice. Můžeme tak například vytvořit funkci nakresli_přechodovou_charakteristiku(čitatel, jmenovatel).

Pro náš konkrétní příklad chceme vytvořit funkci, která umožní zobrazit přechodovou charakteristiku jednoho a více přenosů. Její úplný zdrojový kód je přiložen jako příloha na CD. Výstupem má být obrázek přechodové funkce. Argumenty funkce tedy budou: *jméno souboru*, *rozlišení obrázku* pro funkci print(), které určí jeho výslednou velikost, *čitatel*, *jmenovatel*, *vzorkovací perioda* pro případné diskrétní systémy a *jméno* prvního přenosu. Následovat mohou dále čtveřice hodnot čitatel, jmenovatel, vzorkovací perioda a jméno pro další přenosy. V aplikaci Matlab tedy zapíšeme do souboru m-file deklaraci:

```
function multi_step(filename,dpi,num,den,Ts,jmeno,varargin)
```

Klíčové slovo „varargin“ určuje, že funkce přijme po základní sadě argumentů i libovolný počet argumentů dalších a zpřístupní je za běhu kódu v k tomu speciálně určeném poli. V dalším kódu provedeme kontrolu vstupů, zda je počet prvků pole varargin dělitelný čtyřmi, tedy zda je zadaný správný počet argumentů. V případě že ne, vrátí funkce výjimku s námi definovaným popisem. Tato výjimka bude, stejně jako všechny ostatní výjimky, předána nadřazené aplikaci. To zajišťuje automaticky Matlab Builder při překladu knihovny. Díky tomu také nemusíme provádět typovou kontrolu vstupních argumentů.

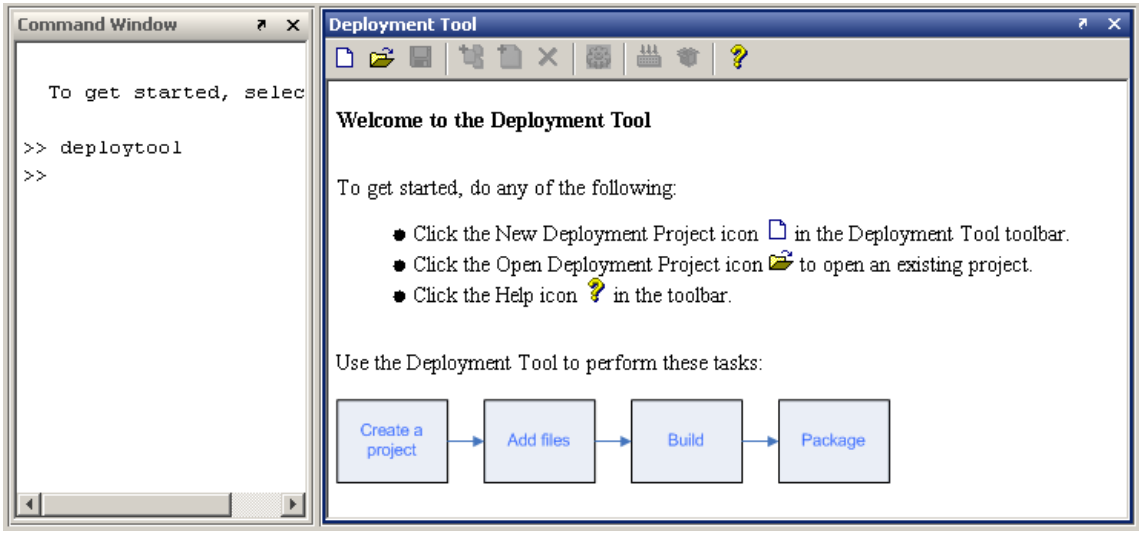
Jádrem funkce je vytvoření přenosů z koeficientů polynomů zadaných v argumentech, vykreslení přechodové charakteristiky těchto systémů a její uložení na místo specifikované proměnnou „filename“. Tato proměnná může obsahovat i absolutní cestu, ne jen název souboru, viz dokumentace funkce print().

Obdobným způsobem vytvoříme i další funkce, které provádějí různé další operace se zadanými systémy. V demonstrační úloze je celkem vytvořeno pět funkcí pro analýzu systémů: *multi_setp*, *multi_nyquist, multi_bode, multi_pzmap* a *multi_rlocus*. Jak vyplývá z názvu, vykreslují přechodovou, Nyquistovu frekvenční a Bodeho frekvenční charakteristiku, polohu pólů a nul a trajektorii pólů. Přijímají naprosto identické argumenty, liší se pouze vykreslovanou charakteristikou. Dále jsou ve vlastní třídě vytvořeny tři funkce pro úpravy přenosů: *diskretizuj, zments* a *spoj*. Tyto funkce slouží pro diskretizaci spojitého systému, přepočet parametrů systému pro změnu vzorkovací periody a pro spojitou aproximaci diskrétního systému.

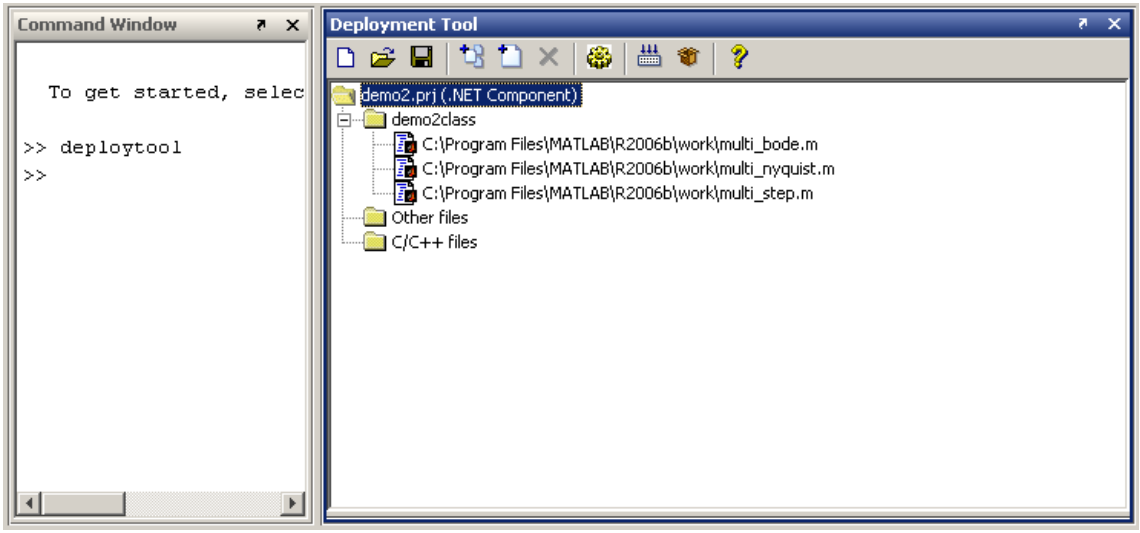
### 4.1.2 Kompilace knihovny

Pro sestavení a kompilaci knihovny použijeme nástroj aplikace Matlab Deployment tool, který spustíme příkazem deploytool.

Vytvoříme nový projekt a z levého menu vybereme Matlab Builder for .NET. Tento builder pracuje se dvěma základními typy knihoven, standardu .NET a standardu COM. V této práci byl použit standard .NET, neboť se jedná o modernější technologii umožňující mimo jiné i snadnější nasazení. Zatímco knihovnu standardu COM je nutné před použitím zaregistrovat v registrech operačního systému, standard .NET umožňuje tzv. xcopy deployment – tedy instalaci pouhým zkopírováním dané knihovny [7]. Tím je možné se vyhnout mnoha problémům s instalací.




*Ilustrace 11: Prostředí nástroje Deployment Tool*




*Ilustrace 12: Základní struktura knihovny pro export*

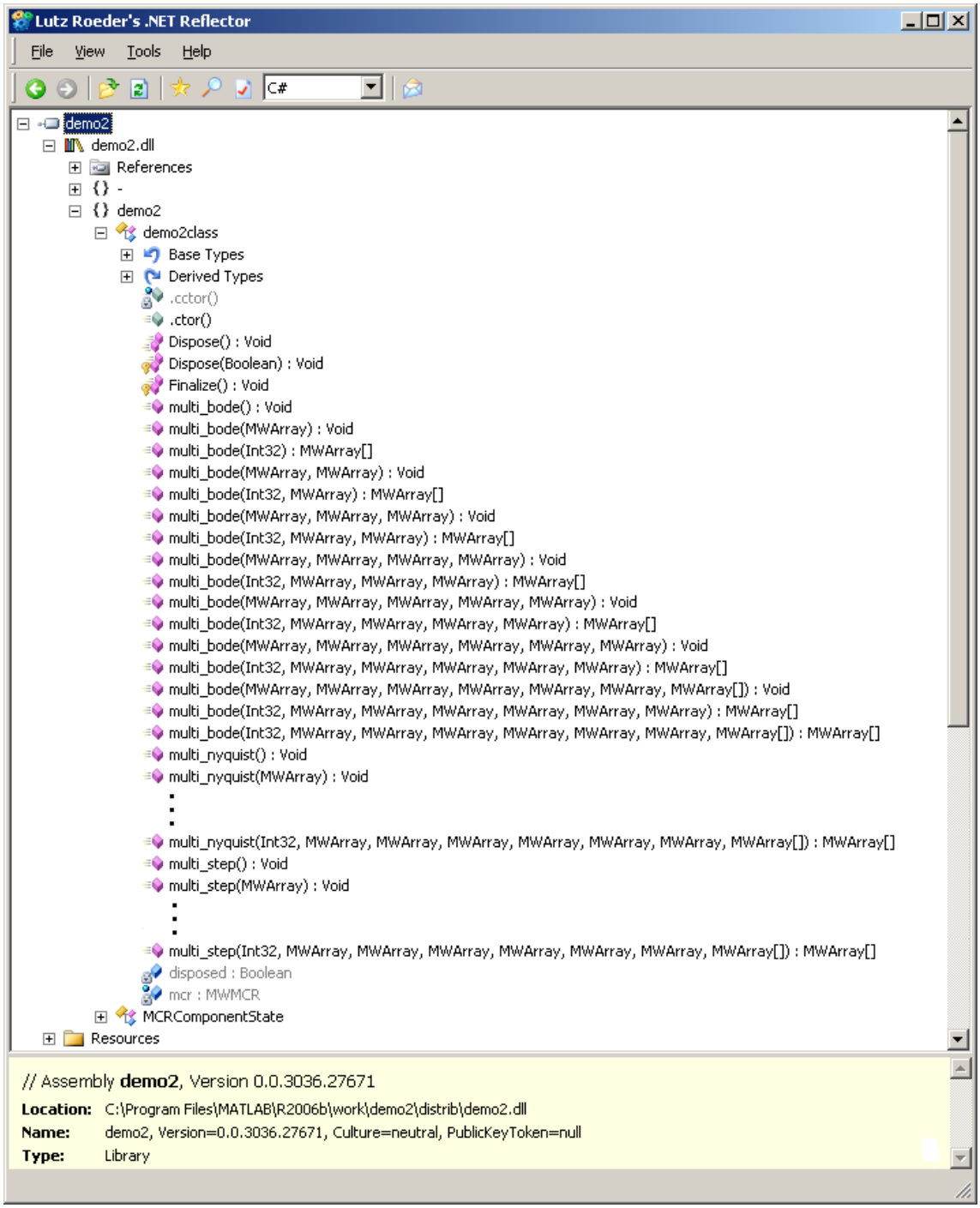
Ve stromové struktuře vidíme první třídu definovanou v naší knihovně, implicitně nese název projektu s přidaným řetězcem „class“. Do ní můžeme přidávat libovolné funkce v podobě souborů m-file, případně můžeme přidat celou další třídu.

Postup doposud popsaný je dobře zdokumentován přímo výrobcem aplikace Matlab. Nicméně na žádné ze tří instalací této aplikace, na kterých byl v průběhu vypracování této práce zkoušen, nefungoval. Byl nuntý následující nezdokumentovaný krok: v nastavení projektu vložíme do pole Compiler Options File soubor *compopts.bat* přiložený k této diplomové práci jako příloha B. Jedná se o soubor nastavení kompilátoru, převzatý z ukázkové úlohy. Tento soubor by pravděpodobně měl být vygenerován při nastavení kompilátoru přímo v aplikaci Matlab a použit automaticky, zřejmě se to však v některých případech neděje.

S takto provedeným nastavením můžeme spustit sestavení knihovny. Výstupem jsou již zmíněné dva soubory: jméno_projektu.dll a jméno_projektu.ctf, nacházející se v nově vytvořeném adresáři jméno_projektu/distrib. Pokud knihovnu vytváříme pro někoho jiného či pro jiný počítač, je vhodné přiložit ještě správnou verzi instalátoru MCR, kterou najdeme v adresáři MATLAB\R2006b\toolbox\compiler\deploy\win32 jako soubor MCRInstaller.exe. Hlavním důvodem je nutnost nainstalovat na cílovém počítači verzi MCR odpovídající verzi aplikace Matlab na které jsme prováděli sestavení.

### 4.1.3 Struktura výsledné knihovny

Použijeme opět program Reflector pro zobrazení struktury knihovny, kterou jsme právě vytvořili.

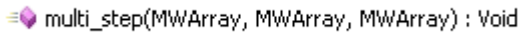


Jak můžeme vidět, tři funkce, které jsme exportovali, se převedly na tři metody třídy, každá s mnohonásobným přetížením. U dvou tříd byla z ilustrace pro zachování přehlednosti odstraněna většina metod. Protože všechny tři funkce akceptují stejné argumenty, liší se i metody pouze názvem. Proto rozebereme jen jednu.

Funkce deklarovaná v aplikaci Matlab jako:

```
function multi_step(filename,dpi,num,den,Ts,jmeno,varargin)
```

přijme šest a více argumentů. Jinými slovy, pole varargin může být prázdné, ale žádný jiný argument nesmí být vynechán. Naproti tomu můžeme vidět v zobrazení struktury definici funkcí, které přijímají jen část z povinných argumentů. Například funkce




deklarovaná jako

```
public void multi_step(MWArray filename, MWArray dpi, MWArray num);
```

přijme i pouhé tři argumenty. Vzhledem k tomu, že nemáme v argumentech obsaženy koeficienty jmenovatele, nemůže vnitřní kód metody fungovat. Pokud funkci takto zavoláme, nenahlásí chybu, ale vrátí prázdný výsledek. Důvod**,** proč při konverzi Matlab Builder for .NET vytváří i tyto varianty s neúplnou deklarací se nepodařilo zjistit, nicméně je potřeba s touto skutečností počítat. Pokud dostaneme od někoho knihovnu bez dokumentace, je třeba si uvědomit, že nepovinný je zpravidla pouze poslední argument pojmenovaný varargin.

Můžeme si také povšimnout, že se všechny varianty funkce vyskytují s dvěma typy návratové hodnoty: void a MWArray[]. Naše funkce sice nemá vracet hodnotu, relevantní jsou tedy jen varianty s návratovou hodnotou void, ale Matlab Builder vygeneruje u každé funkce i návrat hodnoty typu pole. Takto se chová proto, že funkce aplikace Matlab umožňují vracet i různý počet proměnných, dle zadání uživatele. Pokud v aplikaci Matlab například zadáme příkaz

```
[y,t] = step(num, den);
```

předá se funkci step i informace o tom, kolik návratových hodnot je po ní požadováno. V prostředí programovacích jazyků CLS není toto chování standardní,

proto tuto skutečnost reprezentuje v metodě generované Matlab Builderem varianta metody, která jako první argument přijme integer reprezentující požadovaný počet prvků v poli výstupních hodnot. Příkladu výše by tedy odpovídal v programovacím jazyce C# zápis

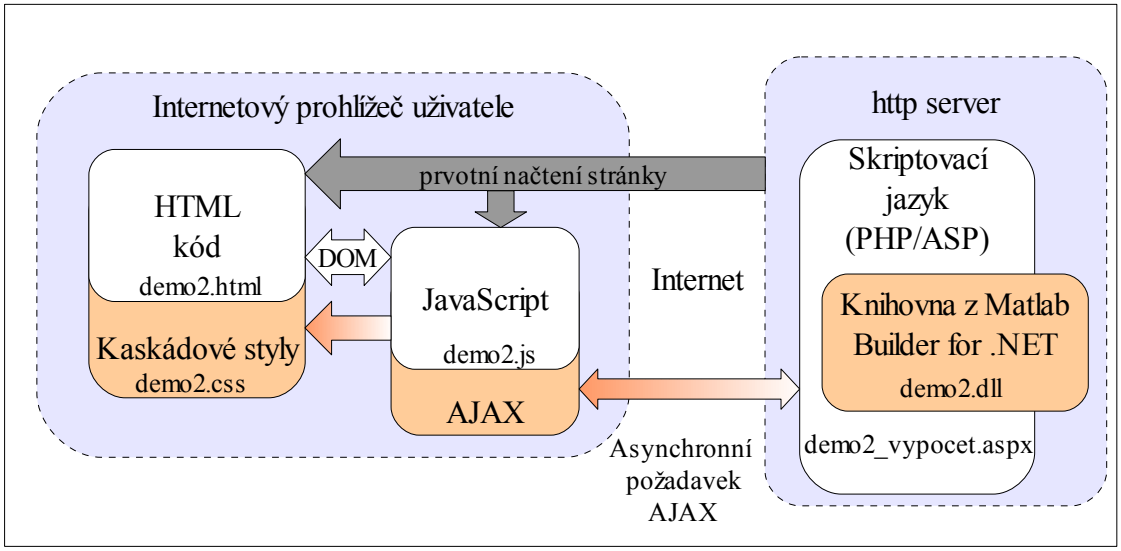
```
MathWorks.MATLAB.NET.Arrays.MWArray[] vysl = vypoc.step(2, num, den);
```

## 4.2 Nasazení knihovny výpočetních funkcí

Na cílovém počítači, v našem případě HTTP serveru, nainstalujeme MCR. Pouze v případě že je již nainstalovaná samotná aplikace Matlab správné verze tento krok není nutný. V takovém případě byla při instalaci aplikace Matlab nainstalována i MCR. Knihovnu budeme využívat jen na HTTP serveru proto, aby uživatel nemusel na svém počítači nic instalovat.

### 4.2.1 Struktura Internetové aplikace

Po Internetové aplikaci podpory výuky požadujeme dynamickou odezvu na akce uživatele, grafické uživatelské rozhraní a přiměřenou rychlost, to vše bez nutnosti instalovat jakákoli rozšíření a programy na straně uživatele. Jako výchozí podmínky pro běh na straně uživatele tedy určíme Internetový prohlížeč podporující DOM level 1, CSS 2.1 a JavaScript. To splňují oba hlavní prohlížeče, Internet Explorer (verze 6 a vyšší) a Mozilla Firefox (verze 2 a vyšší). Tyto požadavky jsou srovnatelné s požadavky jiných internetových stránek.




*Ilustrace 14: Zvolená struktura internetové aplikace*

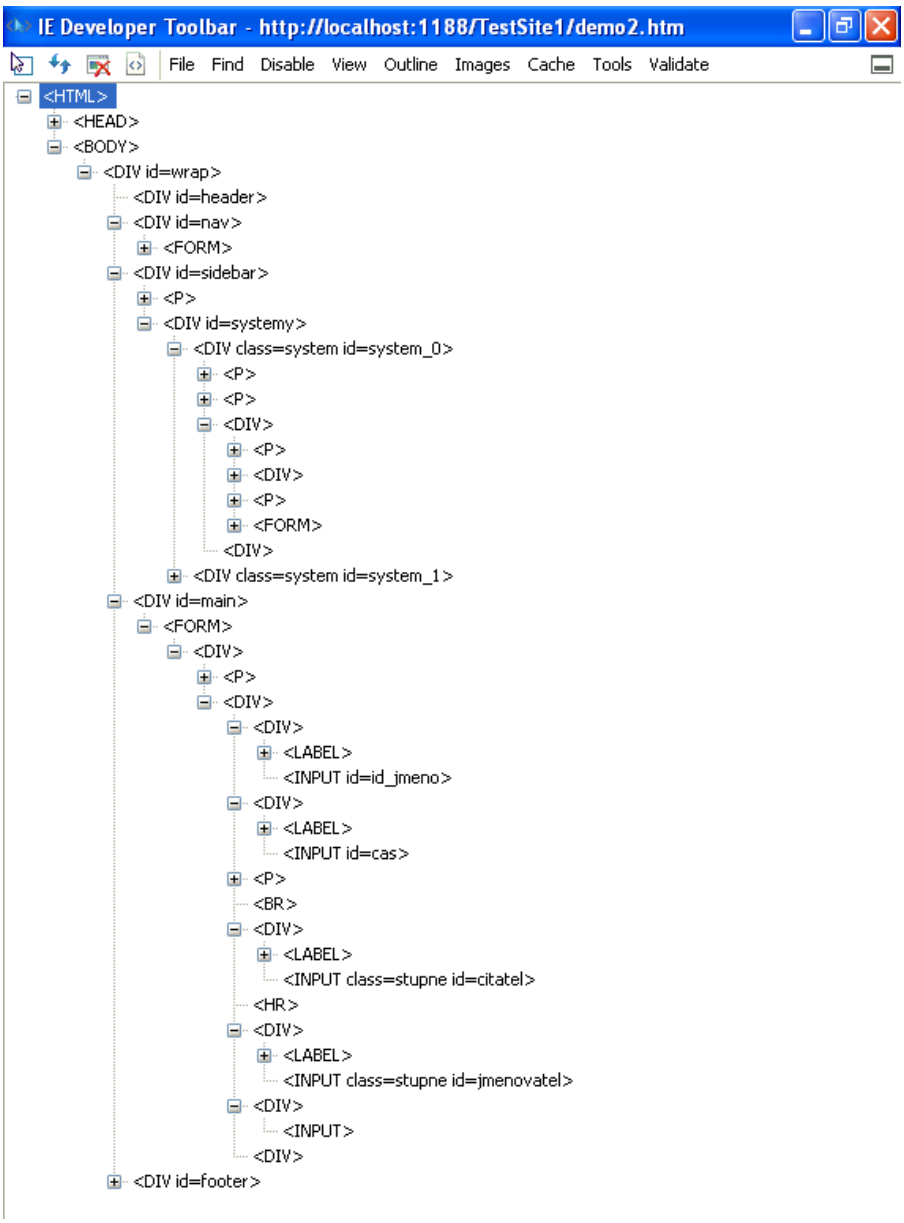
Zvolená struktura internetové aplikace je zobrazena v ilustraci 14. O veškerou interaktivitu, která nevyžaduje výpočet aplikace Matlab, se stará JavaScript ve spolupráci s DOM na straně uživatele. Tím odpadá nutnost komunikovat se serverem a činnost uživatele není omezována rychlostí jeho připojení. Výpočty nutné k zajištění interaktivních funkcí jsou velmi jednoduché a nezatíží neúnosně počítač klienta.

V momentě, kdy je třeba provést výpočet pomocí funkcí exportovaných z aplikace Matlab se prohlížeč obrátí na server pomocí technik AJAX a zažádá o zpracování výpočtu. Ze serveru se vrátí zpráva o ukončeném výpočtu po níž provede JavaScript načtení obrázku výsledku ze serveru. Celková interakce mezi serverem a klientem při této akci obvykle nepřesáhne 10 kB přenesených dat, včetně načtení obrázku. To umožní velmi rychlou odezvu a přispívá k pocitu plynulé interakce.

### 4.2.2 Vzhled dokumentu - HTML, DOM a CSS

Struktura internetových stránek má podobu stromu, obdobně jako struktura formulářové aplikace. Každý grafický prvek (element) má svého předka (parent), který určuje jeho pozici ve výsledném dokumentu. Způsob implementace tohoto stromu a pohybu v něm určuje Document Object Model (DOM). Vzhled jednotlivých prvků dále definují kaskádové styly Cascading Style Sheets (CSS). Ke každému typu prvku či každé třídě je pomocí CSS přiřazen soubor vlastností, které určují jeho vzhled. Celkový

vzhled stránek tak můžeme dynamicky měnit právě zásahem do struktury dokumentu a/nebo kaskádových stylů. Můžeme měnit vlastnosti prvků, mazat je a vytvářet nové, měnit jejich vzhled. Výhodou je, že při jakékoli změně ve struktuře dokumentu provede internetový prohlížeč automaticky jeho překreslení. Ke změně vzhledu stránky tedy stačí vhodným prostředkem zasáhnout do struktury dokumentu a kaskádových stylů. Tímto prostředkem bývá nejčastěji JavaScript, jazyk pro skriptování na straně uživatele.




*Ilustrace 15: DOM struktura stránek podpory výuky automatického řízení*

### 4.2.3 JavaScript

Pro vytvoření dynamických stránek s maximální rychlostí odezvy je nejvhodnějším prostředkem jazyk JavaScript. Je to interpretovaný skriptovací jazyk, jehož kód je vykonáván přímo v prohlížeči uživatele. Umožňuje tak provádět okamžité akce, bez nutnosti kontaktovat HTTP server. V tomto případě DOM a CSS poskytují prostředky a JavaScript vykonává veškeré změny. Do stránek je JavaScript začleněn v podobě externího zdrojového souboru, v našem případě demo2.js. Tento soubor obsahuje kód všech funkcí použitých na stránkách. Tyto funkce zajišťují změny DOM a CSS tak, aby byly akce uživatele doprovázeny odpovídající reakcí stránek – zmáčknutí tlačítka, přidání systému do seznamu vybraných a podobně.

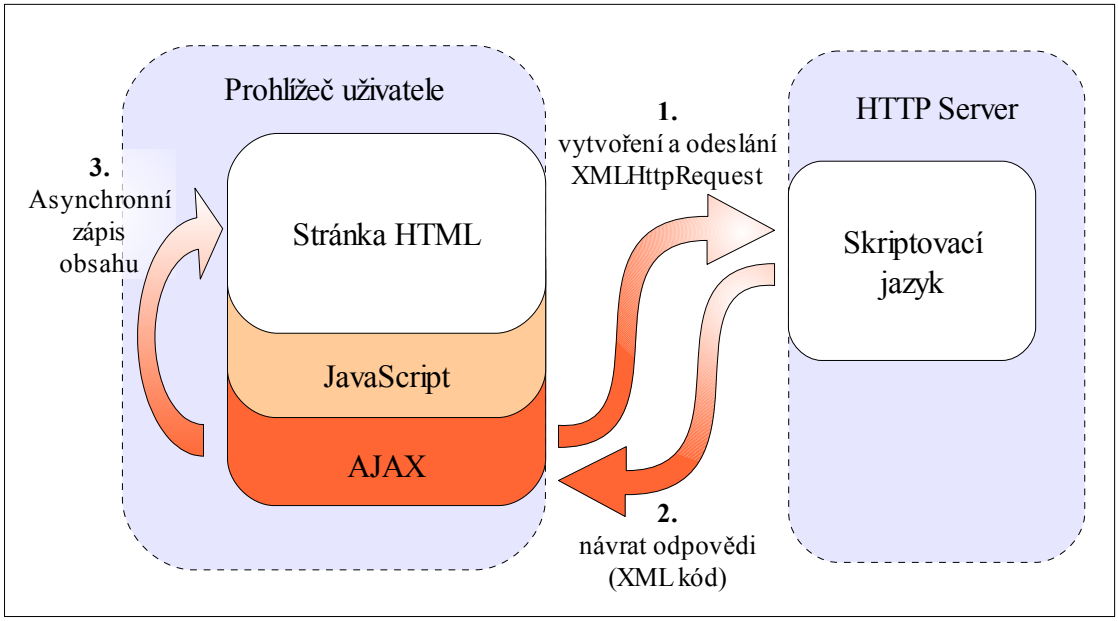
Zdrojový kód JavaScript obsahuje tři globální proměnné. První *isIE* je využita pro určení zda klient používá internetový prohlížeč Internet Explorer (v tom případě je nutné ošetřit některé chyby které obsahuje). S tím souvisí i druhá globální proměnná *pocatek*, která slouží k ošetření rozdílu mezi interpretací struktury DOM různých prohlížečů. Třetí globální proměnná obsahuje opakovaně použitou část kódu HTML.

První funkcí je funkce *init()*. Tato funkce slouží k provedení akcí po dokončení načítání stránky, jako je detekce prohlížeče a předběžné načtení některých obrázků. Následuje funkce *menu_choose(tlacitko)*, která obsluhuje hlavní menu na horní straně stránky. Kdykoli je v tomto menu zmáčknuto tlačítko, provede funkce příslušnou změnu v dokumentu. Funkce *edit_sys(formular,modify)* provádí přidání a změny přenosů zadaných uživatelem. Stará se o změny obsahu levé části stránky – zásobníku systémů. Parametr *modify* říká, zda předané informace určují změnu stávajícího či vytvoření nového systému. Funkce *diak(text)*, *mocniny(pole,znak)* a *parse_pole(pole)* slouží k úpravám, formátování a kontrole uživatelem zadaných dat. Například funkce *diak(text)* odstraňuje z názvu systému diakritiku, která by se ve funkci exportované z aplikace Matlab nezobrazila korektně. Funkce *upravuj_sys(system)* je volána při volbě systému, který chceme upravit a vyplní hlavní oblast stránky potřebnými údaji tak, aby formulář pro změnu obsahoval v současnosti platné údaje. Funkce *smazat(formular)* slouží k případnému vymazání zvoleného systému. Obdobou funkce *uprav_sys(system)* je funkce *analyzuj_sys(system)*. Používá se při volbě systémů, na kterých chceme

provádět analýzu pomocí funkcí exportovaných z aplikace Matlab. Přitom využívá i funkci po ní následující, *porovnej_cisla(a,b)*, která porovnává dvě čísla uložená jako textový řetězec. Předtím, než odešleme na server dotaz kterým žádáme výpočet pomocí funkcí aplikace Matlab, je provedena úprava zasílaných dat pomocí funkce *analyza_choose(typ)*. Tato funkce zakóduje informace o systému do podoby odpovídající datům zaslaným z HTML formuláře pomocí metody *post*. Obdobný proces provede při upravě systému která vyžaduje výpočet aplikace Matlab funkce *zmen_sys(formular, typ)*. Takto upravená data jsou předána funkci *request(data,typ)*, která pomocí techniky AJAX (viz následující kapitola) pošle asynchronní dotaz na server. Obsahem dotazu jsou právě zakódovaná data systému a typ analýzy, kterou chceme provést. Poslední funkcí JavaScript je funkce *zpracuj(data,typ)*, která použije data vrácená serverem a vloží je patřičným způsobem do stránky.

### 4.2.4 AJAX

Pro dosažení maximální plynulosti interakce uživatele s obsahem internetových stránek se postupem doby vyvinuly techniky, jak dosáhnout komunikace mezi internetovým prohlížečem uživatele a serverem bez nutnosti obnovení celé stránky. Pro jednu z těchto technik se vžilo označení AJAX, tedy Asynchronous JavaScript and XML či Asynchronní JavaScript a XML. Jak již bylo zmíněno, jedná se o techniky, nikoli technologie. AJAX využívá stávajících technologií JavaScript a XML pro dosažení specifického cíle. Základní princip ukazuje ilustrace 16.




*Ilustrace 16: Princip funkce techniky AJAX*

Prvním krokem je vytvoření objektu typu XMLHttpRequest. Tento objekt je obdobou požadavku na vrácení stránky, který zasílá prohlížeč při zadání Internetové adresy či při kliknutí na odkaz. Následně použijeme metodu objektu XMLHttpRequest send() a odešleme ho na server spolu s libovolným obsahem. Na serveru se provede zpracování skriptovacím jazykem a zpět je vrácen dynamicky vygenerovaný obsah. Tím může být část HTML kódu, proměnné zapsané ve vhodné XML notaci, či jakýkoli jiný obsah, včetně například binárních souborů. Po příchodu odpovědi do prohlížeče uživatele může být její obsah zpracován pomocí JavaScript a začleněn do stránky. To vše bez nutnosti stránku jako celek obnovovat.

V našich stránkách je technika AJAX využita v JavaScript funkci *request(data,typ)*. Tato funkce vytvoří požadavek pro server a jako jeho tělo předá obsah argumentu data. Následně čeká na událost *onreadystatechange*. Tato událost nastane v okamžiku, kdy se změní vlastnost XMLHttpRequest.readyState. Ta určuje stav požadavku pro server a nabývá pěti hodnot: 0 - objekt ještě nebyl inicializován, nebyla volána metoda open(). 1 - objekt je otevřen pro zápis metodou open(), ale ještě nebyl odeslán metodou send(). 2 - požadavek je odeslán serveru, ale ještě není k dispozici odpověď, 3 - odpověď je právě přijímána. 4 - odpověď serveru je přijata a je k dispozici pro další zpracování; interakce se serverem byla ukončena. Při události *onreadystatechange* provedeme

zjištění, zda je stav *readystate* již roven číslu 4 a požadovaná operace na serveru již byla dokončena. V případě že ano, zkontrolujeme, jaký stavový kód server v odpovědi vrátil. Tento stavový kód slouží k snadné identifikaci, zda požadavek na serveru proběhl bez problému. Nabývá různých hodnot, jejich kompletní přehled můžeme nalézt například v [19]. Hodnota 200 znamená bezproblémové vykonání požadavku. Pokud jsou tedy splněny podmínky XMLHttpRequest.readyState = 4 a XMLHttpRequest.status = 200 proběhl požadavek správně. V našem případě to znamená, že na serveru je připraven obrázek který jsme po něm žádali vytvořit a stačí ho pouze načíst.

### 4.2.5 Pokus o využití knihovny v PHP

Jako první byl proveden pokus využít knihovnu exportovaných funkcí ve skriptovacím jazyce PHP, na němž běží i současné stránky podpory výuky. I když technologie .NET byla původně navržena společností Microsoft, její široké rozšíření vedlo i některé konkurenty k implementaci standardu .NET. Konkrétně v PHP se jedná o třídu *DOTNET*, která slouží k vytvoření instance třídy standardu .NET [8]. Bohužel, tato implementace není úplná a umožňuje použití jen knihoven registrovaných v GAC, neumožňuje tedy xcopy deployment.

Pro umístění komponentu .NET do GAC musí mít tento komponent globálně unikátní jméno, tzv. strong name. To obsahuje navíc oproti klasickému jménu knihovny digitální podpis, vygenerovaný z privátního klíče tvůrce [9]. Tato skutečnost s sebou přináší řadu komplikací. Matlab Builder for .NET dokáže vytvořit komponentu s unikátním jménem, nicméně k tomu potřebuje privátní klíč. Ten je třeba vytvořit pomocí programu obsaženého v .NET framework SDK, který však k ničemu jinému nevyužijeme; pro vytvoření knihovny potřebujeme jen samotný .NET framework. Další komplikace nastávají při instalaci knihovny na cílovém počítači, neboť již nestačí pouhé zkopírování na potřebné místo, ale je potřeba komponentu umístit do GAC pomocí k tomu určeného programu či instalátoru. Nejzávažnější komplikací je to, že popsané řešení je v rozporu s principy .NET a GAC. Xcopy deployment byl do standardu .NET zaveden zvláště proto, aby jednoúčelové knihovny využívané pouze v jedné aplikaci nemusely být umístěné v globálním úložišti a programátoři se tak vyhnuli problémům typu DLL Hell.

Z výše popsaných důvodů se zdá, že čas pro nasazení standardu .NET v PHP ještě nenastal. Nicméně neustálý vývoj PHP snad v dohledné době umožní přidat plnou podporu .NET, včetně načtení assembly ze samostatného souboru.

### 4.2.6 Využití knihovny v ASP.NET

Po neúspěchu s PHP byl pro ověření funkce a demonstraci nasazení knihovny z Matlab Builder for .NET použit skriptovací jazyk ASP.NET. Tento produkt firmy Microsoft je přímo postaven na technologii .NET a měl by tedy být schopen bez problémů s knihovnou spolupracovat. Technologie ASP.NET je, stejně jako vývojové prostředí Microsoft Visual Web Developer Express Edition, zdarma [10], [11]. K nasazení ASP.NET tedy potřebujeme jen verzi operačního systému, která obsahuje IIS, což je HTTP server firmy Microsoft. Tyto podmínky splňuje například operační systém Microsoft Windows XP Professional.

Prvním krokem při využití knihovny je určení, které komponenty bude internetová stránka využívat, což je rozhodující při kompilaci zdrojových kódů ASP. Jedná se o obdobu externích referencí aplikace psané například v C++. Toto nastavení provedeme v souboru web.config našich stránek přidáním řádky ve tvaru:

```
<add assembly="MWArray, Version=1.0.2415.7148, Culture=neutral,
PublicKeyToken=e1d84a0da19db86f"/>
```

do sekce assemblies. Parametr mezi uvozovkami je úplné jméno assembly, které můžeme například snadno získat pomocí aplikace Reflector zmíněné v kapitole 2.3.1. Tato řádka říká, že chceme použít assembly daného jména. Standardně ASP.NET tuto assembly hledá v GAC a v adresáři BIN. ASP.NET je stejně jako celý .NET framework citlivý na velikost písma (case-sensitive) [12], rozeznává tedy, na rozdíl od operačního systému Windows, velká a malá písmena. V adresáři „bin“ tedy knihovna nebude nalezena.

Dalším krokem je napsání samotného ASP skriptu, který konečně využije programový kód exportovaný z aplikace Matlab. Jeho úplný zdrojový kód je přiložen jako příloha na CD. Prvním využitím části aplikace Matlab je vytvoření pole proměnných typu MWArray, který je součástí MCR:

```
MathWorks.MATLAB.NET.Arrays.MWArray[] argumenty = new
MathWorks.MATLAB.NET.Arrays.MWArray[(pocet_systemu_navic) * 4];
```

Jedná se o odkaz na prostor jmen MathWorks.MATLAB.NET.Arrays a v něm obsaženou třídu MWArray. Toto pole použijeme k uložení dynamického pole argumentů, které se bude předávat naší funkcí jako argument typu varargin, viz kapitola 4.1.1.

Následující řádek pak využívá již námi vytvořenou knihovnu:

```
demo2_2.analyza pocitadlo = new demo2_2.analyza();
```

Opět se odkážeme na prostor jmen a v něm obsaženou třídu. Vytvoření této instance znamená automaticky spuštění výkonného jádra MCR a od této chvíle je exportovaný kód připraven k využití v podobě metod instance třídy. V další části kódu vytvoříme ještě proměnné typu MWNumericArray a MWCharArray použité k uložení parametrů předávaných volané metodě. Je vidět i použití implicitního přetypování obsaženého taktéž v knihovně MWArray.dll, například na řádku:

```
MathWorks.MATLAB.NET.Arrays.MWCharArray soubor =
string.Concat(Server.MapPath("TEMP"), "\\bdn");
```

je vytvořena matice znaků z proměnné typu string pomocí implicitní konverze. V argumentu, který představuje jméno výsledného obrázku je vhodné uvést celou cestu, čímž je zaručeno jeho správné umístění. V opačném případě se obrázek vytvoří kdesi hluboko v adresářové struktuře systému, kde je spuštěno prostředí MCR.

Samotné volání metody a tedy vykonání funkce aplikace Matlab je provedeno řádkem

```
pocitadlo.multi_step(soubor, dpi, num, den, Ts, jmeno, argumenty);
```

kterým voláme metodu multi_step() objektu pocitadlo s parametry, které jsme v předchozím kódu vytvořili. Odkazujeme se tak na tuto definici:

```
public void multi_step(MWArray filename, MWArray dpi, MWArray num,
MWArray den, MWArray Ts, MWArray jmeno, params MWArray[] varargin);
```

obsaženou v naší knihovně dll. Všechny parametry musí být typu odvozeného od typu MWArray. Poslední argument, proměnná „argumenty“, je pole proměnných odvozených od typu MWArray. Je možné použít přímo dynamicky generované pole jako v tomto

případě nebo, pokud je to výhodnější, můžeme pokračovat v zápisu argumentů oddělených čárkou. Libovolné argumenty následující po argumentu „jmeno“ budou předány cílové funkci v poli varargin.

Tímto voláním se provede výpočet a uložení obrazu grafu do souboru určeného argumentem „filename“. Nyní stačí sdělit prohlížeči uživatele, že je pro něj obrázek připraven a může si ho stáhnout.

Pokud provádíme vývoj skriptu ASP v prostředí Microsoft Visual Web Developer Express Edition, je třeba dát si pozor na chybu při nakládání s dočasnými soubory. Pokud nahrajeme do adresáře našeho projektu novou verzi dll knihovny a změníme patřičný odkaz v souboru web.config, Visual Studio neprovede korektně zkopírování druhého souboru knihovny s příponou .ctf do adresáře dočasných souborů [20]. Tato chyba se projeví výjimkou System.NullReferenceException při pokusu o volání metody instance třídy z knihovny. Řešením je celý dočasný adresář smazat, pak se při spuštění správně překopírují oba potřebné soubory.

### 4.2.7 Výsledná aplikace a její výkon

Protože ASP.NET využívá JIT kompilaci, může první spuštění skriptu trvat až kolem třiceti sekund. Nicméně po prvním spuštění již program pracuje značnou rychlostí, plně srovnatelnou s rychlostí výpočtu v samotné aplikaci Matlab. Při testu na lokálně umístěném serveru, tedy bez zdržení vlivem přenosu dat po síti Internet, se doba od zmáčknutí tlačítka uživatelem po zobrazení výsledného obrázku pohybuje na počítači vybaveném procesorem Intel Centrino Core Duo T2050 1,6 GHz, 512 MB RAM a operačním systémem Windows XP Professional od jedné do dvou sekund. Protože při použití technik AJAX nedochází k obnově celé stránky, je objem přenesených dat menší. Lze tedy předpokládat, že výsledná rychlost bude záviset hlavně na výpočetní složitosti funkcí exportovaných z aplikace Matlab, tedy že doba výpočtu na serveru bude hlavním limitujícím faktorem.

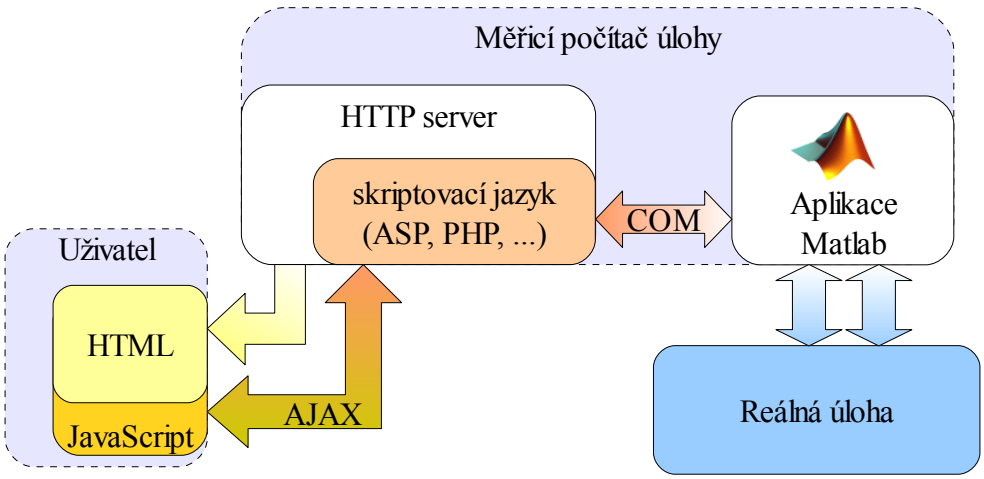
Tato demonstrační internetová aplikace slouží pouze pro ověření konceptu a dokumentaci základního postupu využití Matlab Builder for .NET. Proto nesplňuje některé požadavky na dobře napsanou internetovou stránku. Splnění těchto požadavků

by zabralo značné množství času, bez konkrétního přínosu k tématu této diplomové práce. Zvláště se jedná o to, že není možné uložit stav aplikace, tedy ani uživatelem vytvořené systémy (to je řešitelné pomocí cookies nebo databáze na serveru), a o nefunkčnost stránky bez zapnutého JavaScript. Funkčnosti bez JavaScript je možno dosáhnout za cenu ztráty rychlosti: všechny funkce JavaScript, které aplikace obsahuje, by bylo třeba realizovat na straně serveru ve skriptovacím jazyce ASP. Pokud prohlížeč uživatele nezpracuje JavaScript, odeslala by se potřená data na server, který by vygeneroval změněný obsah stránek. Pro tento algoritmus je připraven soubor *no_js,aspx*, na který odkazují všechny formuláře. Pak by se místo zapsání do DOM provedlo obnovení celé stránky. Tím získá uživatel plně funkční, i když nikoli tak efektní a komfortní stránky. V současném stavu také aplikace obslouží vždy jen jednoho uživatele, neboť soubor obsahující obraz grafu má konstantní jméno, neustále se tedy přepisuje. Řešením je generovat jméno náhodné a sdělit ho prohlížeči uživatele pomocí AJAX. Zároveň bude třeba zabránit hromadění souborů na serveru prováděním údržby adresáře, kam jsou obrazy ukládány. Na stránkách také nebyla provedena validace standardu W3C.org, i když jsem se maximálně snažil jej dodržet.

Pokud se stávající stránky použijí jako forma, je možné velmi rychle a snadno doplnit další funkce. Stačí rozšířit stávající knihovnu o další metody a ty pak implementovat v ASP skriptu, případně přidat podle postupu popsaného v kapitole 4.2 knihovnu další. Tento krok nezabere zkušenému uživateli více než dvacet minut.

# 5 Možnosti realizace vzdáleného měření

Původním záměrem bylo využít Matlab Builder for .NET i k vytvoření podmínek vzdáleného měření v e-laboratoři spojitého řízení. Při postupném seznamování se s možnostmi tohoto nástroje se ukázalo, že toto řešení zřejmě není perspektivní. Hlavní překážkou je to, že Matlab Compiler a potažmo Matlab Builder for .NET nespolupracuje s Matlab Simulink, který ovšem tvoří jádro softwarové podpory úlohy. Spojení měřicích karet a aplikace Matlab zprostředkovává Real Time Toolbox, který je rozšířením právě Matlab Simulink. Bez tohoto prostředku není možné ve stávající hardwarové konfiguraci propojit aplikaci Matlab s reálnou úlohou. V současné době se proto zdá výhodnější inovovat stávající návrhy vzdáleného měření a řízení, například návrh popsaný v [16]. Matlab Web Server je možno nahradit libovolným HTTP severem, který bude nainstalován přímo na měřicím počítači úlohy a s aplikací Matlab spojen například pomocí rozhraní COM. Protože při tomto uspořádání bude server obsluhovat pouze uživatele který právě provádí měření, nezatíží ani příliš měřicí počítač. Využitím moderních technik návrhu internetových stránek, které v posledních letech prodělaly rychlý vývoj (zejména techniky AJAX), můžeme dosáhnout dalšího zkvalitnění vzdáleného měření a řízení bez nutnosti zásadně měnit uspořádání úlohy.




*Ilustrace 17: Návrh úpravy stávající koncepce vzdáleného měření a řízení*

# Závěr

Většinu cílů této diplomové práce se podařilo splnit. Při průzkumu možností komunikace aplikace Matlab s ostatními programy byly zjištěny některé nové skutečnosti, jako je omezení podporovaných funkcí pro produkty závislé na Matlab Compiler či možnost ovládat aplikaci Matlab přes rozhraní standardu COM. Zřejmě nejpřínosnější v této části bylo vytvoření dokumentace popisující fungování a způsoby použití Matlab Builder for .NET. Při tvorbě této dokumentace byly odhaleny některé potenciální problémy, které mohou nastat při nasazení Matlab Builder. Zároveň popis fungování a podmínek použití Matlab Builder for .NET usnadní případné nasazení v jiných aplikacích.

Podařilo se vytvořit základ funkčních internetových stránek podpory výuky, které využívají funkcí aplikace Matlab zakomponovaných do programu na HTTP serveru. Umožňují uživateli vytvořit model systému ve tvaru obrazového přenosu a ten pak dále upravovat, diskretizovat spojité modely či naopak provést spojitou aproximaci diskrétního systému. K dispozici je řada funkcí pro analýzu systémů jako je vykreslení přechodové charakteristiky či Nyquistovy frekvenční charakteristiky.

Tyto stránky jasně ukazují široké možnosti, které se otevírají nasazením aplikace Matlab, s jejím takřka neomezeným výpočetním potenciálem, do prostředí Internet. Nyní je dle potřeby výuky a vyučujících možné velmi rychle vytvořit funkce, které budou demonstrovat přednášenou látku či kontrolovat výsledky cvičné úlohy. Přidání takovéto funkce do internetových stránek je otázkou maximálně desítek minut.

Nepodařilo se zrealizovat podmínky pro vzdálené měření a řízení v e-laboratoři spojitého řízení. Ukázalo se, že prostředky, na které se tato práce především soustředí, tedy .NET Framework a Matlab Builder for .NET, jsou pro takové nasazení za podmínek daných současnou koncepcí e-laboratoře zcela nevhodné. Byly tedy alespoň navrženy možné cesty k vylepšení stávajících návrhů vzdáleného měření a řízení bez nutnosti zasáhnout do stávající struktury e-laboratoře.

**Seznam použité literatury**


[1] *COM: Component Object Model Technologies.* [online]
URL: <http://www.microsoft.com/com/default.mspx>

[2] RUFFALDI, Emanuele. *1..2..3 ways of integrating MATLAB with the .NET.* [online] The Codeproject.
URL: <http://www.codeproject.com/KB/dotnet/matlabeng.aspx>

[3] *Options for Deploying MATLAB Applications via the Web*. [online] Mathworks Products & Services URL:
<www.mathworks.com/products/new_products/webserver_discontinued.html>

[4] *Working with the MCR*. [online] MATLAB Compiler Documentation. URL:
<http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/f12-999353.html>

[5] ROEDER, Lutz. *Lutz Roeder's Programming.NET.* [online]
URL: <http://www.aisto.com/roeder/dotnet/>

[6] *How can I load a figure generated by a MATLAB Builder for .NET 2.1 (R2006b) component in an ASP.NET application?* [online] Mathworks Support Technical Solutions
URL: <http://www.mathworks.com/support/solutions/data/1-2EETRY.html>

[7] *Deployment in .Net*. [online] Visual Builder.com Dotnet Tutorials.
URL: <http://www.visualbuilder.com/dotnet/tutorial/pageorder/20/>

[8] *PHP:DOTNET.* [online] PHP Manual
URL: <http://php.morva.net/manual/en/class.dotnet.php>

[9] *Strong-Named Assemblies.* [online] .NET Framework Developer's Guide
URL: <http://msdn.microsoft.com/en-us/library/wd40t7ad.aspx>

[10] *The Official Microsoft ASP.NET Site.* [online] URL: <http://www.asp.net/>

[11] *Visual Web Developer 2008 Express Edition.* [online]
URL: <http://www.microsoft.com/express/vwd/>

[12] *Case-sensitive code.* [online] SitePoint Blogs
URL: <http://www.sitepoint.com/blogs/2005/03/24/case-sensitive-code/>

[13] *MWArray Class.* [online] URL:
<http://www.kxcad.net/cae_MATLAB/toolbox/dotnetbuilder/MWArrayAPI/HTML/MathWorks.MATLAB.NET.Arrays.MWArray.html>

[14] *How can I compile an assembly object using MATLAB Builder for .NET version 2.0 for compatibility with Microsoft .NET Framework 2.0?* [online] The Mathworks Support Technical Solutions URL:
<http://www.mathworks.com/support/solutions/data/1-2K2SPM.html?product=MN&solution=1-2K2SPM>

[15] *1608 - Web Deployment of MATLAB® Applications Guide.* [online] The Mathworks Support Technical Notes
URL: <http://www.mathworks.com/support/tech-notes/1600/1608.html>

[16] Vyšohlíd, J. *Implementace platformy Matlab pro vzdálené řízení v e-laboratoři TK4.* Diplomová práce. TU Liberec, Liberec 2006

[17] Hubka, L. *Implementace informačních a komunikačních technologií v e-laboratoři spojitého řízení.* Diplomová práce. TU Liberec, Liberec 2005

[18] *The Mathworks Product List.* [online]
URL: <http://www.mathworks.com/products/product_listing/index.html>

[19] *status Property (IXMLHTTPRequest).* [online] Microsoft .NET Framework Developer Center library
URL: <http://msdn.microsoft.com/en-us/library/ms767625(VS.85).aspx>

[20] *Why to I receive a "System.NullReferenceException" in Microsoft Visual Studio .NET 2003 when calling a component built with MATLAB Builder for .NET 2.1 (R2006b)?* [online] The Mathworks Support Technical Solutions
URL: <http://www.mathworks.com/support/solutions/data/1-3RWK5L.html>

[21] *Java SE Downloads.* [online] Sun Developer Network
URL: <http://java.sun.com/javase/downloads/index.jsp>

[22] *MATLAB® Compiler™ Unsupported Functions.* [online] MATLAB Compiler Documentation
URL: <http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/bqrvu87-6.html>

[23] *MATLAB® Compiler™ 4.8 Support for MATLAB and Toolboxes.* [online] URL: <http://www.mathworks.com/products/compiler/compiler_support.html>

[24] *Requirements for the MATLAB® Builder™ NE Product - Limitations and Restrictions.* [online] MATLAB® Builder™ NE Documentation URL: <http://www.mathworks.com/access/helpdesk/help/toolbox/dotnetbuilder/ug/bqigjk4-1.html#bqigjk4-4>

[25] *Stats > Browser Trends.* [online] Browser News URL: <http://www.upsdell.com/BrowserNews/stat_trends.htm>

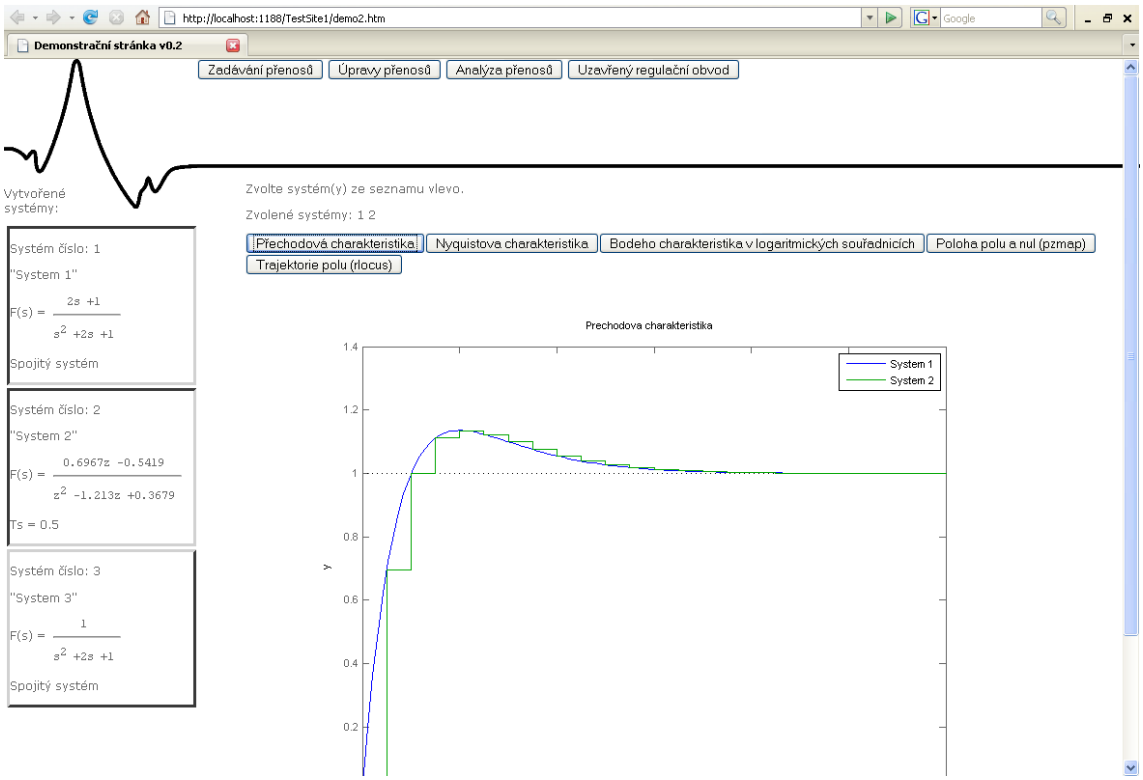
[26] *java vs. C# speed comparisons?* [online] LtU Classic Archives URL: <http://lambda-the-ultimate.org/classic/message5561.html>

[27] *The Great Win32 Computer Language Shootout.* [online] dada's perl lab URL: <http://dada.perl.it/shootout/index.html>

[28] *Java performance.* [online] Wikipedia, the free encyclopedia URL: <http://en.wikipedia.org/wiki/Java_performance>

[29] *C Standalone Application Target.* [online] MATLAB Compiler Documentation URL: <http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/f7-996249.html#f7-995909>

## Příloha A – vzhled výsledné internetové aplikace




*Ilustrace A.1: Vzhled internetové aplikace v prohlížeči Mozilla Firefox*


*Ilustrace A.2: Vzhled internetové aplikace v prohlížeči Internet Explorer*

## Příloha B – soubor compopts.bat

```
@echo off
rem MSVC60COMPP.BAT
rem
rem    Compile and link options used for building MATLAB compiler programs
rem    with Microsoft Visual C++ compiler version 6.0
rem
rem    $Revision: 1.18.4.7 $  $Date: 2006/06/23 19:04:18 $
rem
rem ********************************************************************
rem General parameters
rem ********************************************************************
set MATLAB=%MATLAB%
set MSVCDir=C:\Program Files\Microsoft Visual Studio\VC98
set MSDevDir=%MSVCDir%\..\Common\msdev98
set PATH=%MSVCDir%\BIN;%MSDevDir%\bin;%MATLAB_BIN%;%PATH%
set INCLUDE=%MSVCDir%\INCLUDE;%MSVCDir%\MFC\INCLUDE;%MSVCDir%\ATL\INCLUDE;
%INCLUDE%
set LIB=%MSVCDir%\LIB;%MSVCDir%\MFC\LIB;%LIB%
set PERL="%MATLAB%\sys\perl\win32\bin\perl.exe"
set MW_TARGET_ARCH=win32

rem ********************************************************************
rem Compiler parameters
rem ********************************************************************
set COMPILER=cl
set OPTIMFLAGS=-O2 -DNDEBUG
set DEBUGFLAGS=-Zi -Fd"%OUTDIR%%MEX_NAME%.pdb"
set CPPOPTIMFLAGS=-O2 -DNDEBUG
set CPPDEBUGFLAGS=-Zi -Fd"%OUTDIR%%MEX_NAME%.pdb"
set COMPFLAGS=-c -Zp8 -G5 -GX -W3 -nologo
set CPPCOMPFLAGS=-c -Zp8 -G5 -W3 -nologo -Zm500 -GX -MD -I"%MATLAB
%\extern\include\cpp" -DMSVC -DIBMPC -DMSWIND
set DLLCOMPFLAGS=-c -Zp8 -G5 -GX -W3 -nologo -DMSVC -DIBMPC -DMSWIND
set NAME_OBJECT=/Fo

rem ********************************************************************
rem Library creation commands creating import and export libraries
rem ********************************************************************
set DLL_MAKEDEF=type %BASE_EXPORTS_FILE% | %PERL% -e "print \"LIBRARY
%MEX_NAME%.dll\nEXPORTS\n\"; while (<>) {print;}" > %DEF_FILE%
```

```
rem ************************************************************************
rem Linker parameters
rem MATLAB_EXTLIB is set automatically by mex.bat
rem ************************************************************************
set LIBLOC=%MATLAB%\extern\lib\win32\microsoft
set LINKER=link
set LINKFLAGS=kernel32.lib user32.lib gdi32.lib advapi32.lib oleaut32.lib
ole32.lib /LIBPATH:"%LIBLOC%" /nologo
set LINKFLAGS=%LINKFLAGS% mclmcrrt.lib
set CPPLINKFLAGS=
set DLLLINKFLAGS= %LINKFLAGS% /dll /implib:"%OUTDIR%%MEX_NAME%.lib" /def:
%DEF_FILE%
set HGLINKFLAGS=sgl.lib
set LINKOPTIMFLAGS=
set LINKDEBUGFLAGS=/debug
set LINK_FILE=
set LINK_LIB=
set NAME_OUTPUT="/out:%OUTDIR%%MEX_NAME%.exe"
set DLL_NAME_OUTPUT="/out:%OUTDIR%%MEX_NAME%.dll"
set RSP_FILE_INDICATOR=@

rem ************************************************************************
rem Resource compiler parameters
rem ************************************************************************
set RC_COMPILER=rc /fo "%OUTDIR%%RES_NAME%.res"
set RC_LINKER=

rem ************************************************************************
rem IDL Compiler
rem ************************************************************************
set IDL_COMPILER=midl /nologo /win32 /I "%MATLAB%\extern\include"
set IDL_OUTPUTDIR= /out "%OUTDIRN%"
set IDL_DEBUG_FLAGS= /D "_DEBUG"
set IDL_OPTIM_FLAGS= /D "NDEBUG"
set POSTLINK_CMDS1=if exist %LIB_NAME%.def del %LIB_NAME%.def
```