EVOLUCE VÝVOJE SOFTWARE V PROSTŘEDÍ MICROSTATION

Stanislav Sumbera

stanislav@sumbera.com

Abstrakt
Příspěvek se zabývá jednotlivými programovacími jazyky a vývojovými platformami, které byly do prostředí CAD systému MicroStation postupně zaváděny (MicroCSL, UCM, MDL, MicroStation Basic, Java/JMDL, VBA, VC++, .NET) . U každého vývojového prostředí jsou rozebrány výhody i nedostatky a vliv na vývojářskou komunitu MicroStation. Příspěvek se zaobírá mimo jiné i fenoménem programovacího jazyka JMDL (hybridní jazyk Javy a C++) a vystihuje budoucí směr vývoje software v prostředí MicroStation.

1. Úvod

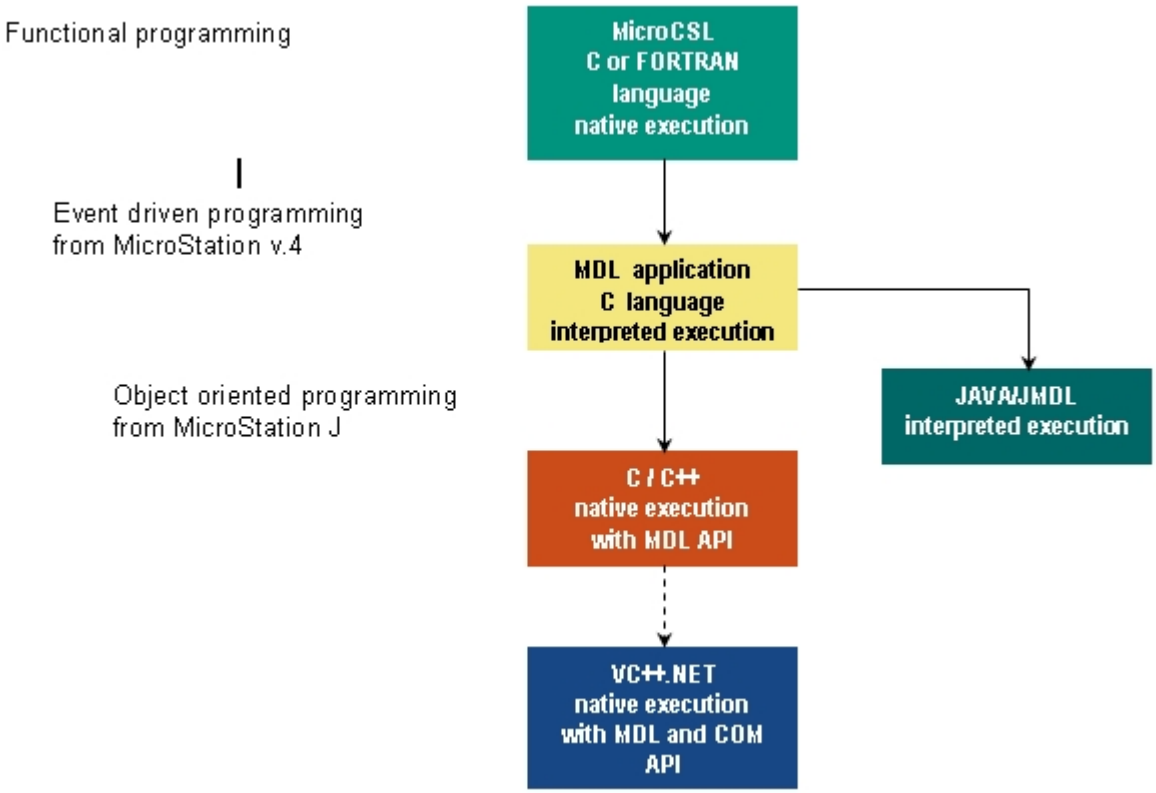
Vývojem aplikací pro MicroStation se zabývá celá řada subjektů v ČR, především v oblastí Geoinformačních systémů. CAD systém MicroStation tedy není třeba blíže představovat. MicroStation prošel a stále prochází řadou inovací v oblasti vývoje aplikací. Ne vždy ovšem byly inovace pro vývojáře přínosné a často vkládaly do vývoje v prostředí MicroStation pochyby o tom, kterým směrem MicroStation bude směřovat a zda-li investice do přeškolení programátorů na nový jazyk, resp. platformu bude přínosem v dlouhodobém výhledu. Pokusíme se v jednotlivých kapitolách zhodnotit jednotlivé stádia ve vývoji softwaru pro systém MicroStation.

V MicroStation můžeme vývoj softwaru rozdělit do dvou hlavních řad:

1. Vývojová řada jazyka C
2. Vývojová řada jazyka Basic

2. Vývojová řada jazyka C

V této vývojové řadě se drží hlavní vývoj pro prostředí MicroStation. Následující obr 1 schématicky znázorňuje jednotlivé jazyky a platformy, přerušovanou čárou je nastíněn možný budoucí směr. V následujících kapitolách se budeme stručně jednotlivými jazyky a jejich zvláštnostmi zabývat.

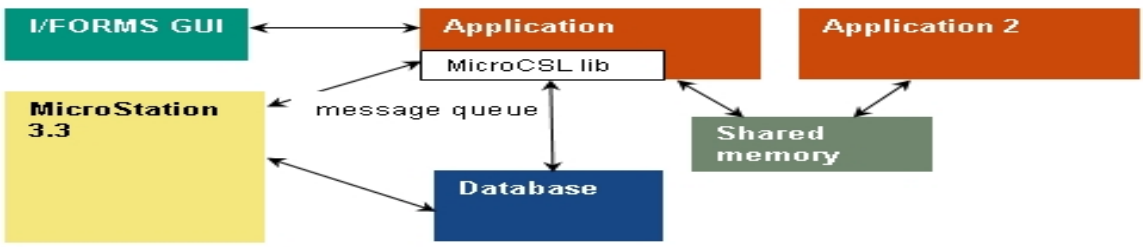


Obr 1. Vývojová řada jazyka C

2.1 MicroStation Customer Support Library - MicroCSL:

MicroCSL byla první vývojovou knihovnou pro MicroStation 3.3. Obsahovala statickou knihovnou pro jazyky C a Fortran s funkcemi pro přímou manipulaci s grafickými elementy v dgn souboru a funkcemi pro práci s databází. Nespornou výhodou MicroCSL byla kompilace kódu do nativních instrukcí procesoru, které byly vhodné především pro časově náročné úlohy. Nevýhodou MicroCSL bylo především sekvenční programování, neúplná integrace s MicroStation a cena. V současné době není MicroCSL podporováno, je plně nahrazeno funkcemi MDL.

Následující obr. 2 ilustruje architekturu aplikací pro MicroStation 3.3 na Unixové platformě.



Obr 2. Architektura aplikací MicroCSL na Unixové platformě

2.2 MicroStation Development Language – MDL

MDL - vývojový jazyk pro MicroStation byl poprvé představen v roce 1991 v MicroStation v. 4. jako jednotné řešení pro vývoj profesionálních aplikací v MicroStation. Bentley zamýšleli vytvořit hlavní vývojovou platformu, která by pokryla jak vývoj jednoduchých utilit, tak vývoj robustních systémů postavených na fundamentální funkčnosti MicroStation. Dalším důvodem vzniku MDL byla platformová nezávislost kódu, kterou se podařilo udržet až do verze MicroStation SE.

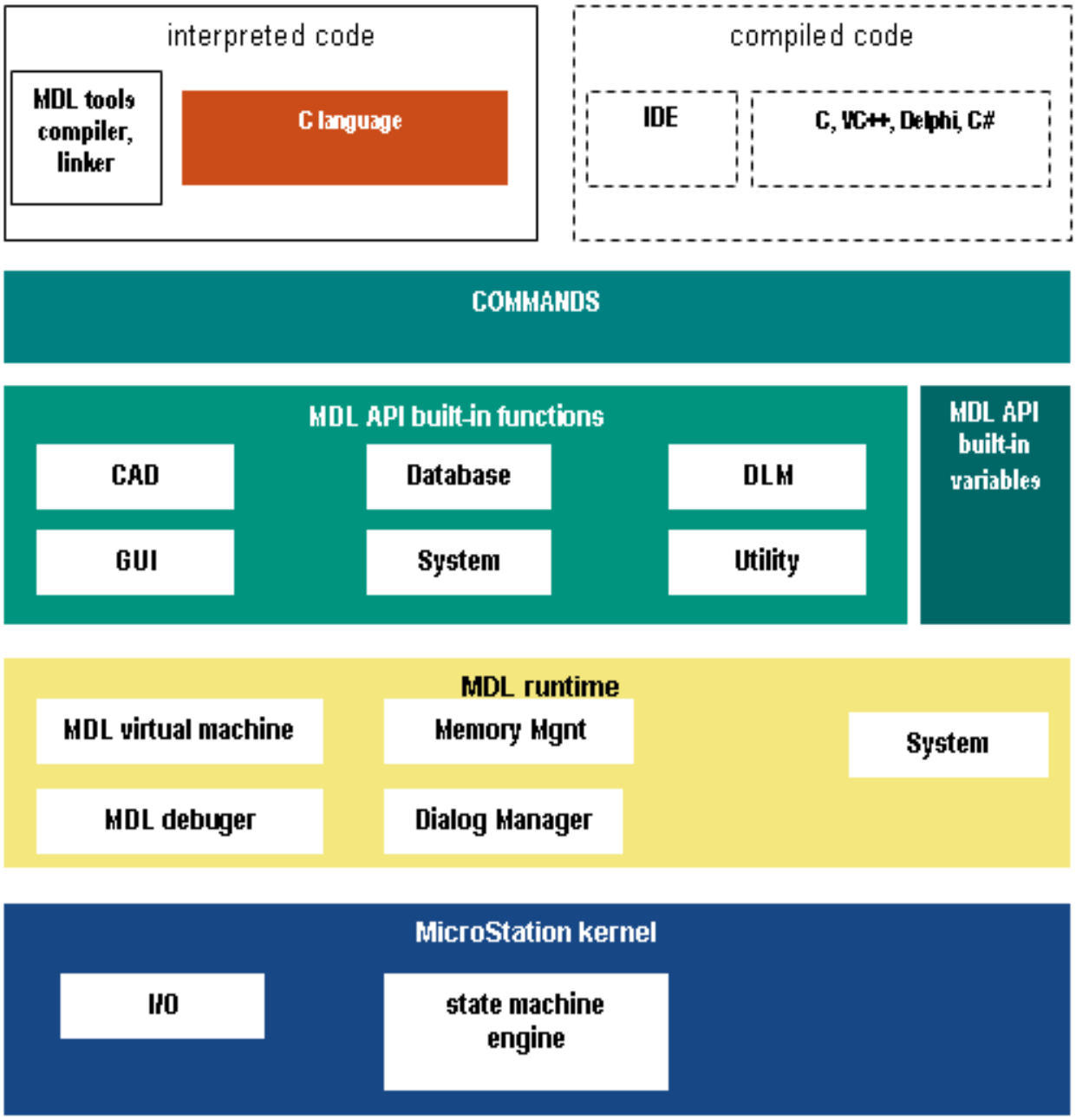
MDL není de-facto žádný speciální jazyk, jak by mohl název naznačovat – je to ANSI C sdrobnými úpravami, který se tradičně překládá dodávaným kompilátorem a linkerem do pseudo kódu, který je v prostředí MicroStation interpretován. Tato vlastnost byla důležitá z hlediska snadné portability kódu. Mezi hlavní nevýhody takto interpretovaného kódu byla a stále je menší rychlost zpracování oproti nativnímu kódu (přestože API MicroStation je z převážné části v nativním kódu) a izolovanost kódu od API poskytovaného operačním systémem. Na druhé straně výhodou může být odstínění programátora od detailů a specialit operačního systému a větší bezpečnost kódu.

I když bylo možné již od verze 5 MicroStation využívat vedle interpretovaného MDL také nativních dynamicky linkovaných knihoven (DLL), situace se radikálně s příchodem poslední verze 8 změnila. Nyní je umožněno program napsaný v MDL s drobnými úpravami přímo překládat do nativního kódu na platformě Windows. To s sebou nese obrovské výhody pro vývojáře – především možnost využít integrovaného vývojového prostředí a nativního debuggeru (např. MS Visual Studio) a dále možnost využít knihoven a komponent poskytovaných pro operační systém Windows.
Klíčovou částí MDL stále zůstává programové rozhraní (API), které čítá řádově tisícovky funkcí.

Celé MDL tedy můžeme brát spíše jako vývojovou platformu pro MicroStation, než jen pouhý jazyk. V MDL je také napsána většina základních utilit pro základní konfiguraci MicroStation.

Architektura MDL

Podívejme se na architekturu MDL pro pochopení souvislostí mezi programovacím jazykem MDL, infrastrukturou MDL, MDL rozhraním, MDL runtime a MicroStation.



Obr.3 Architektura vývojové platformy MDL

Na úplně nejnižší úrovni se nachází základní infrastruktura pro chod MicroStation a MDL aplikaci. Do této části spadají manažery paměti, dialog boxu a základní systémové funkce MicroStation. Tyto funkce jsou interně využívány funkcemi MDL API a běžný programátor k nim nemá přístup.

Nad touto úrovní se nachází MDL API tvořící řádově stovky funkcí, které jsou dokumentované a přístupné programátorům v MDL. Mezi MDL API patří také sada proměnných. Nad MDL API se nachází vrstva příkazů (commands), které lze volat prostřednictvím tzv. key-in (vstupní příkazy operátora) přímo z MicroStation. Každý příkaz je navázán na určitou funkci v kódu MDL. Poslední vrstva je samotný programovací jazyk :

- interpretovaný MDL – tj. ANSI C, který pro své spuštění vyžaduje MDL runtime. Pro překlad kódu do pseudo-kódu používá MDL vlastní

kompilátor a linker. Pro debuggování je nutné využít vestavěný symbolický řádkový MDL debugger

- nativní kód (šrafovaně) . Teoreticky je možné psát kód využívající MDL API v libovolném jazyce a vývojovém prostředí, podporujícím vytvoření nativní dynamické knihovny (DLL). Výhodou je především možnost použít libovolného jazyka (např. psát kód objektově v C++), integrované vývojové prostředí a v neposlední řadě vizuální debbuger splňující současné nároky programátorů.

2.3 Java/JMDL:

MicroStation /J s sebou přinesla integrovanou Javu a JMDL - derivát Javy a C++ . Ačkoli začlenění Javy bylo logickým krokem na tehdejší moderní Javovou vlnu, JMDL se setkal s de-facto neúspěchem. Namísto aby Bentley vytvořili čistě Javovské třídy pracující s výkresy a MicroStation, vytvořily tyto třídy ve vlastním jazyku JMDL. JMDL byl původně určen pro tzv. ECM – Engineering Component Model, od kterého Bentley prozatím ustoupili. Využití JMDL lze pozorovat v MicroStation Geographics iSpatial, kde je využito Javy pro komunikaci s Oraclem a JDML pro zobrazování výkresů, resp. v produktu Project Bank. JMDL je nadmnožinou jazyka Java a obsahuje mimo jiné podporu struktur, ukazatelů, šablon, preprocesoru jazyka C, přímé volání MDL nebo DLL kódu (obdobné J/Direct). S příchodem MicroStation v.8 nedošlo k žádným vylepšením ani publikací tříd zahrnutých do JMDL, došlo k povýšení Java virtuálního stroje z verze 1.1.8. na Javu 1.2. Namísto toho se v Microstation v8 objevilo nové ‘objektové’ prostředí pro programování v MicroStation - Visual Basic for Application v.6. a nová COM komponenta s názvem ‘MicroStationDGN’. JMDL je bezesporu zajímavý pokus o překonání problému Javy a C++ a sloučení jejich dobrých vlastností do jednoho jazyka, nicméně díky slabé podpoře ze strany Bentley, nedokumentovaným funkcím (dokumentovány byly pouze třídy pro práci s výkresem DGN), nedokonalou vzájemnou komunikací mezi MDL kódem a JDML kódem zůstává tento inovační krok v pozadí celkového vývoje v systému MicroStation.

2.4 VC++/MFC

Teprve s verzí 8 přichází pro vývojáře plná podpora nativního vývoje aplikací v prostředí MicroStation. Bentley na podzim roku 2002 také uvolnili knihovny pro podporu psaní uživatelského rozhraní v MFC. Tento krok je bezpochyby správným a nezbytným rozhodnutím, protože MicroStation už ztratil svou platformovou nezávislost a upnul se na operační systém Windows – tím by tedy měl plně využívat všech jeho možností. Vývojářům se tak přímo otevírá cesta pro využívání MDL API pouze pro práci s výkresy DGN a ovládání MicroStation, zatímco pro práci s databází, uživatelské rozhraní nebo interoperabilitu mohou využít jakékoli jiné (vhodnější) API. Další informace viz. [2],[5].

2.5 VC++.NET

Lze předpokládat, že vývoj pro MicroStation na platformě Windows půjde v nejbližších letech směrem .NET aplikací. .NET v současné době poskytuje velmi solidní podporu interoperability jak s COM objekty tak s DLL kódem – což je důležité pro MDL API funkce, které jsou z dynamicky linkovaných knihoven přístupné. Je tedy možné už dnes psát in-process aplikace pro MicroStation v .NET. Jendou s klíčových rolí sehrává technologie IJW (It Just Work), která umožňuje transparentně provolávat nativní kód z kódu .NET, viz. [1],[4]. Příklad smíšeného kódu:

```
private: System::Void button1_Click(System::Object * sender, System::EventArgs * e){
   mdlSystem_newDesignFile("design.dgn");  //... otevření výkresu
}
```

3. Vývojová řada jazyka Basic

3.1 MicroStation User Command Language – UCM

UCM byl navržen jako makro jazyk simulující postup uživatele. Jazyk využíval především bohaté základny příkazů MicroStation, prostřednictvím kterých umožňoval manipulaci s grafickými elementy ve výkresu. Jeho výhodou byla jednoduchost a snadná čitelnost kódu. Nevýhodou naopak primitivní syntaxe, pouze sekvenční programování, rychlost a nedokonalé uživatelské rozhraní.

3.2 MicroStation Basic

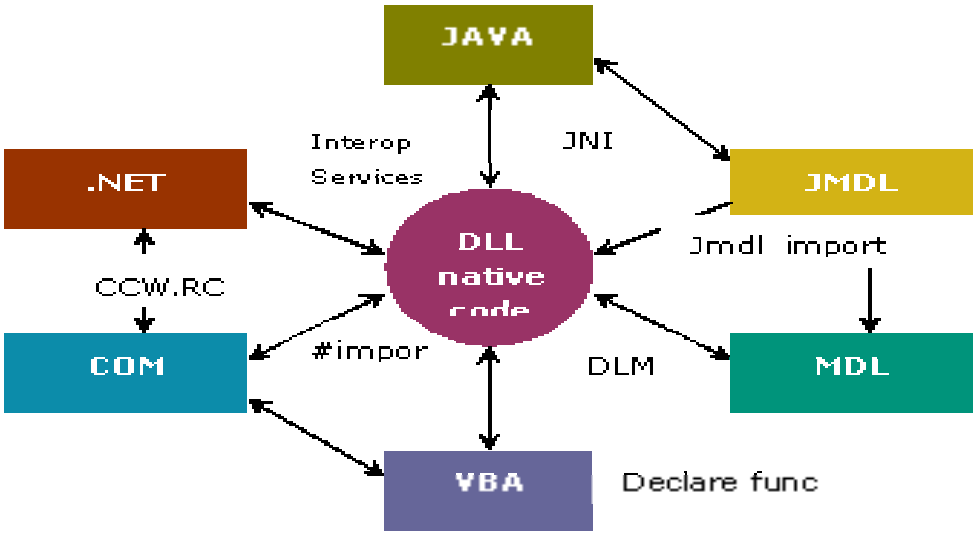
MicroStation Basic byl dalším virtuálním strojem nasazeným od verze MicroStation 95, který měl nahradit stávající UCM. Byl z části napsán v jazyku MDL, obsahoval jednoduchý editor a vizuální debugger. Byl výbornou pomůckou pro jednoduché skriptování, práci s výkresem, kustomizaci prostředí a jiné jednodušší operace. Jednou z nevýhod byly modální dialogová okna a omezená délka kódu. Přestože využíval syntaxe známé z Visual Basicu, nebyl s tímto jazykem plně kompatibilní, neumožňoval zavádění komponent, přímé volání MDL kódu apod. Naopak jednou z výhod byla možnost nahrání uživatelských postupů. MicroStation Basic si našel ve vývojářské komunitě MicroStation své místo a stal se velmi oblíbeným pro svou jednoduchost.

3.3 Visual Basic for Application 6

MicroStation v8 s sebou přináší plně integrovaný Visual Basic for Application (VBA)licencovaný společností Microsoft v takové podobě jaký jej známe např. z produktu Office. Pro tento vývojový nástroj bylo také vytvořeno nové API v podobě COM objektu MicroStationDGN. VBA samozřejmě umožňuje integraci ActiveX komponent, provolávání nativního kódu apod. VBA je bezesporu moderním řešením vývoje pro systém MicroStation, nicméně přichází v době, kdy je nasazována technologie nové generace - .NET a s tím také nové řešení pro vývoj aplikací - Visual Studio for Application (VSA).

4. Závěr

Jednou s klíčových rolí každé nové vývojové platformy je interoperabilita se stávajícím řešením. Poslední obr. 4 tohoto článku zobrazuje interoperabilitu mezi vybranými jazyky v prostředí MicroStation. Ústředím této interoperability je nativní DLL knihovna, která může být napsaná tradičně ve Win32, ale také ve Visual C++ a nebo nejlépe ve Visual C++.NET.



Obr. 4. Interoperabilita programových prostředí v MicroStation

Literatura:

1. Sumbera, S. 2003. Using VC++.NET for MicroStation v8 programming. Nebublikováno. http://www.sumbera.com/ustation/research/vcnetbox.htm
2. Sumbera, S. 2003. Setting up Visual Studio IDE for MDL programming. ControlAltDelete, Pen & Brush publishers Australia, No. 1, 2003, pp. 22-25.
3. Sumbera, S. 2002. Developing raster management with Oracle 9i and MicroStation v8. ControlAltDelete, Pen & Brush publishers Australia, No. 3, 2002, pp. 20-23.
4. Sumbera, S. 2002. .NET & VBA interoperability in MicroStation v8. ControlAltDelete, Pen & Brush publishers Australia, No. 2, 2002, pp. 36-40.
5. Sumbera, S. 2002. Writing MFC based dialogs for MicroStation v8. MicroStation Manager. Bentley System Inc. USA. No. 01, 2002, pp. 36-37.
6. Sumbera, S. 2001. Java/JMDL communication with MDL applications. MicroStation Manager. Bentley System Inc. USA. No.12, 2001, pp.30-34.