Windows 8 – Metro Style- Runtime – WinRT

Thomas Reinwart

Mit dem ab Oktober 2012 ausgelieferten Windows 8 zieht eine neue zusätzliche Benutzeroberfläche **Metro** ein, die wohl weitreichendste Änderung von Microsoft seit dem Umstieg von DOS auf Windows. Es ist der Versuch, eine einheitliche flüssige und moderne Oberfläche auf PC, Tablets, Mobile und - was auch immer die Zukunft bringen mag - weiteren Geräte zu etablieren. Dahinter verbirgt sich viel mehr – nämlich die *Windows Runtime (WinRT)*, auf denen Metro Applikationen aufbauen.

Windows RT

Erstmalig in der Unternehmensgeschichte waren die letzten Quartalszahlen negativ, der Druck ist hoch, nun gilt es mit etwas neuem - Windows 8, WinRT und Metro wieder einen Erfolg zu erzielen. Um das Vorhaben von Microsoft, Windows 8 / WinRT / Metro am Markt zu etablieren und damit erfolgreich zu werden, zu unterstreichen, hat Microsoft vor kurzem einen eigenen Tablet PC mit dem Namen *Surface* vorgestellt und geht damit auch ein Risiko ein, seine bisherigen Hardware Partner zu verärgern. Der Prozessor ist nicht mehr auf Intel sondern auf ARM Basis, das Design bunt, es geht wohl auch in die Richtung sich vom Apple Hardware Markt einen Anteil zu holen.

WinRT wird von Microsoft als Bestandteil des Betriebsystems gesehen, nicht als zusätzliches Framework. WinRT wird daher auch mit jeden Build von Windows kompiliert. Es werden die Prozessorarchitekturen von x86, x64 und ARM unterstützt.

Architekturunterschiede zwischen .net und WinRT

Die in den letzten Jahren bisher am meisten genutzte Art von Entwicklungsumgebung auf Windows Systemen ist .net. Vor .net verwendete jede Programmiersprache seine eigene Laufzeitumgebung, dementsprechend schwierig war es, wenn Programmteile unterschiedlicher Programmiersprachen miteinander arbeiten sollten.

.net verwendet die *CLR (common language runtime)*, bei der es sich um eine Laufzeitumgebung handelt die .net Code ausführt und Dinge wie *Memory Management* zur Verfügung stellt. Mit *Common* ist gemeint, dass in der Laufzeitumgebung mehrere Programmiersprachen (C#, VB.net, ...) unterstützt werden. Beim Code handelt sich um *managed code*. Der Compiler erzeugt keinen Maschinencode mehr wie früher, sondern eine *Intermediate Language (IL)*, die objektorientiert ist. Erst der *just in time compiler (JIT)* übersetzt den *IL code* in die Maschinensprache die für den Computer passt. Inzwischen entstand *DLR (dynamic language runtime)*, einem Aufsatz auf der CLR, die für die dynamischen Sprachen wie Python, Ruby und JavaScript konzipiert ist.

Das .net Framework bietet in seiner Version 4.0 etwa 14.000 Klassen an, einem umfangreichen bequemen Zugang zur Bibliothek.

Als Kritikpunkte von .net gelten das stiefmütterliche behandelte C++, das auf der Liste der weltweit verwendeten Sprachen neben Java noch immer hoch im Kurs steht. Zwar gibt es hier eine spezielle CLR Variante C++/CLI, diese hat sich aber nicht durchgesetzt. Durch die Verwendung von JIT dauert der allererste Aufruf eines .net Programmes auf einer Maschine immer etwas länger. Als weitere Punkt der *Garbage Collector*, der für das Aufräumen der nicht mehr benötigten Objekte im Speicher zuständig ist, dieser stellt eine Bequemlichkeit für die Entwickler dar, weil man sich nicht um das Aufräumen verwendeter Objekte um Speicher kümmern muss. Allerdings bei ressource-kritischen Anwendungen wie Bildbearbeitung und Spielen eine nicht ausreichende Kontrolle über den Aufräumvorgang zulässt. Weiteres kommt die Interoperabilität zwischen .net und anderen nicht CLR Sprachen nur durch *PInvoke (Plattform Invoke)* zustande, einer Mischform zwischen *managed* und *unmanaged Code*.

Mit WinRT soll das alles besser werden. Aber warum zwei *runtimes* parallel auf dem System?

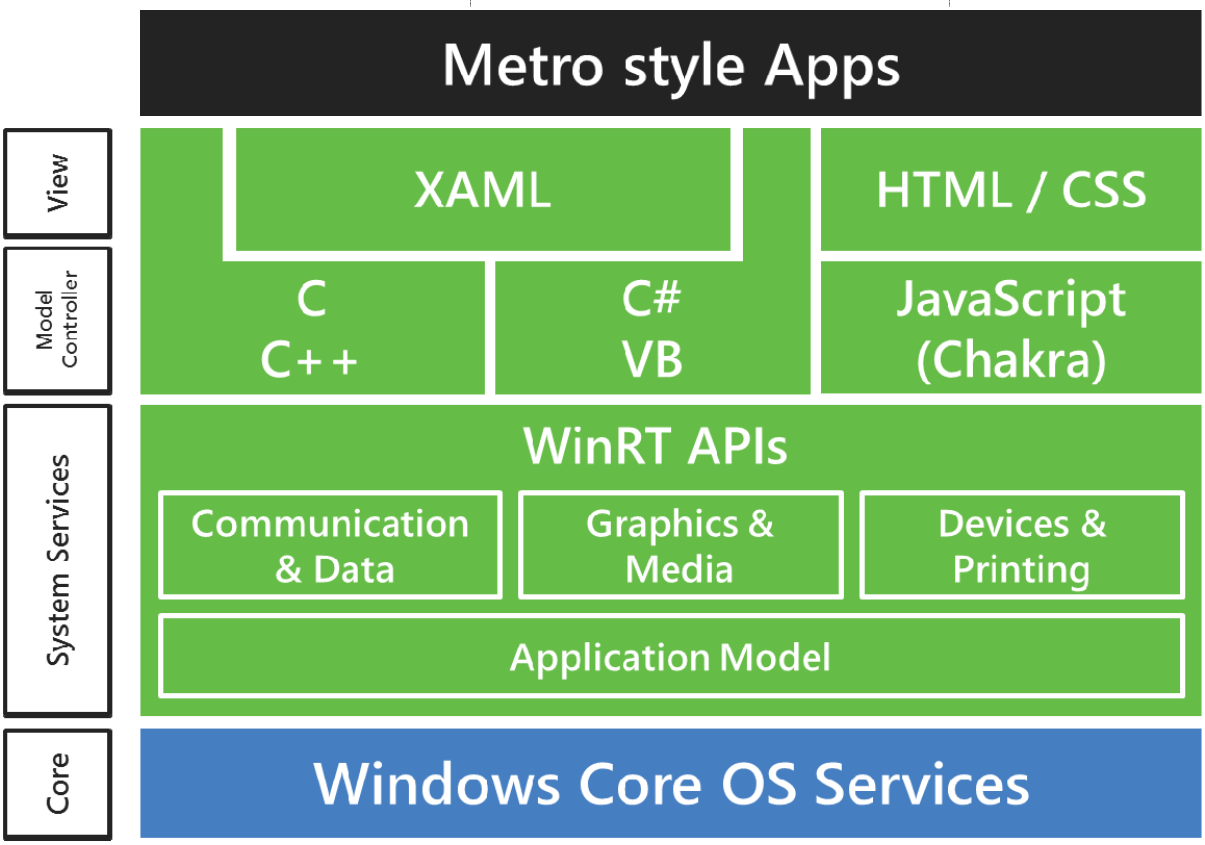
Bei WinRT handelt sich nicht um eine zweite CLR. Microsoft hat für WinRT einen binären Standard *(Application Binary Interface – ABI)* definiert, die eine Komponente einhalten muss, um mit WinRT kompatible zu sein. Eine WinRT Komponente kann weiterhin die CLR nutzen, die *JavaScript Engine (Chakra)* nutzen die WinRT-kompatibel ist, oder C++, oder ... Hauptsache WinRT-kompatibel muss es sein. Somit besteht auch die Wahrscheinlichkeit, dass Programmiersprachen von anderen Herstellern als Microsoft auf den Markt kommen werden, die WinRT kompatible Komponenten erzeugen können.

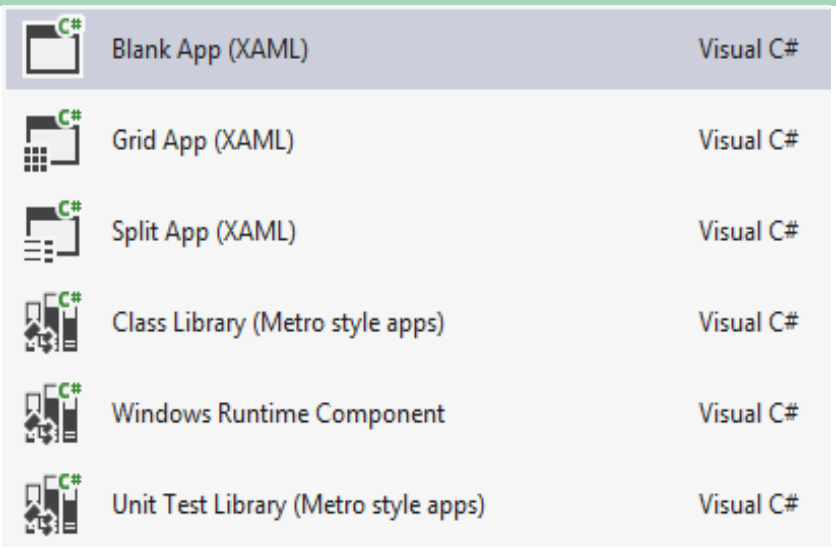
In WinRT sind Konzepte eingeflossen, die sich schon bei Windows und .net bewährt haben. Zum einen *COM (Component object model)* – das es schon sehr lange gibt, und von .net das Metadatenmanagement.

Wir erinnern uns: .net sollte COM im Jahr 2002 ablösen. Seit 10 Jahren war ein Stillstand bei der Entwicklung von COM eingekehrt. COM liefert HRESULT Werte anstatt der Exceptions.

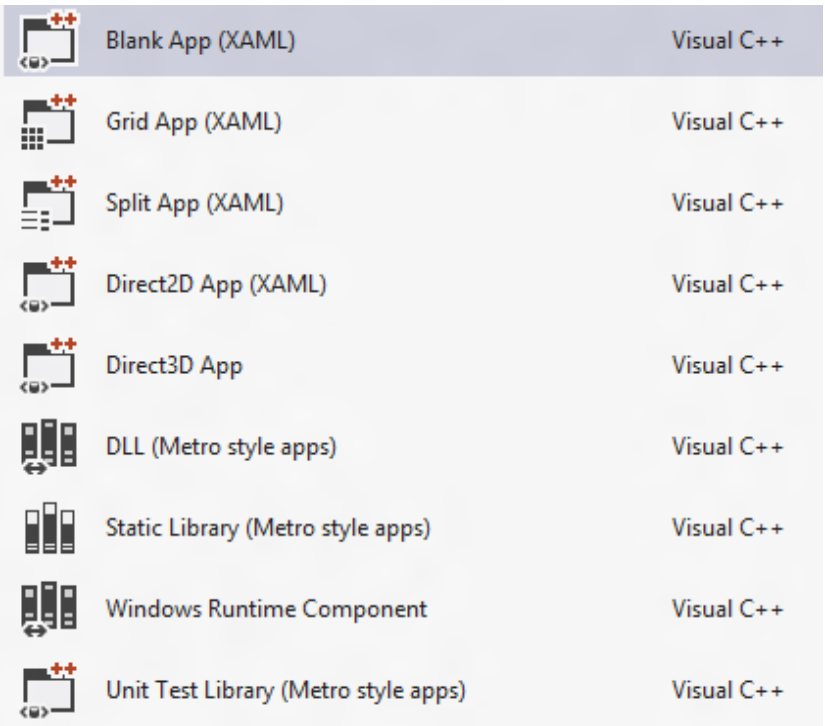
WinRT Komponenten stellen ihre Typen in `.winmd` Dateien zur Verfügung. Der Umfang der WinRT Klassenbibliothek ist zugegebener Weise mit etwa 1.800 Klassen im Vergleich zu .net 4.0 mit 14.000 Klassen bescheiden. Zusätzlich zu *PInvoke* und COM stellt WinRT auch WinRT kompatible Objekte zur Verfügung, durch den viele Teile der .net Klassenbibliothek unnötig sind. Selbst entwickelte WinRT Komponenten sind COM kompatible, das WinRT auf einer neuen Version vom COM aufbaut. COM ist aber wesentlich vereinfacht worden gegenüber früher, keine Type Library mehr sondern eine wincmd Datei.

Die nächste .net Version 4.5 ist für eine Zusammenarbeit mit WinRT ausgelegt, also WinRT Komponenten in .net Komponenten einzubinden oder auch eigene WinRT Komponten zu erstellen.

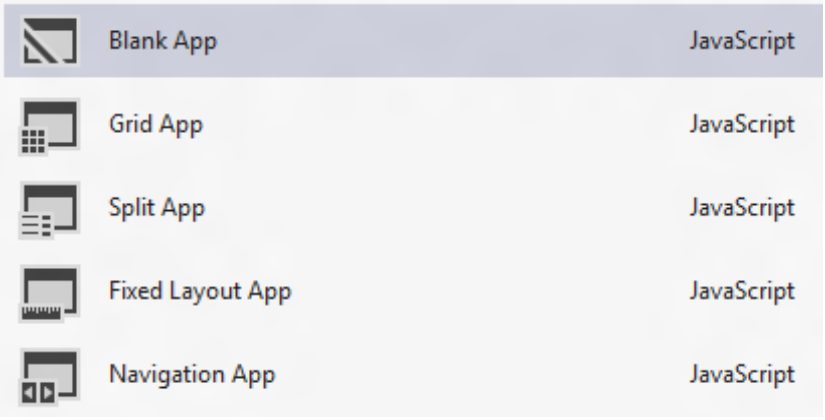




Die VB und C# Templates sind ident



Hier kommen Direct2D/3D hinzu



Programmierung von Metro Apps

Möglich ist dies mit C++/CX, C#, Visual Basic, JavaScript, sogenannte Projektionen (Typisierungen). Einige setzen auf XAML (wie schon aus .net bekannt), die aber durch WinRT ohne *WPF (Windows presentation foundation)* und Silverlight auskommt. Auch DirectX ist möglich. Bei JavaScript ist dies HTML 5. WinRT ist die Ablösung der Win32-Api, also kein Aufsatz auf die Win32-Api wie das .net framework.

WinRT Klassen besitzen Metadaten so wie .net Anwendungen, bei WinRT heißen das Metadatenformat WinMD, das Format dieser Files ist das selber wie bei .net für die CLI, kann somit mit ILDASM oder *Reflector* gelesen werden. Diese findet man unter `Windows\System32\WinMetadata`. *Reflection* (Abfrage der Metadaten zur Laufzeit) wird unterstützt. WinRT ziehlt auf maximale Leistung aber bei geringem Overhead.

Visual Studio 2012 bietet Templates im Windows Metro Style für die Sprachen Visual Basic, C#, C++ und JavaScript an.

Windows 8 App Store

Mit dem Release von Windows 8 können Entwickler ihre Apps über den Microsoft Windows Store vertreiben. Dazu benötigt man ein Konto, die App muss einem Qualitätsprozess durchlaufen, um Schadsoftware, Betrug etc. vorzubeugen. Nicht nur als Firma, auch als Einzelperson kann mein Apps vertreiben. Der Mindestpreis einer App liegt bei 1.49 $, aber auch kostenlose oder zeitlich begrenzte Apps können in den App Store gestellt werden. Auch die Integration von Werbung in die App ist möglich (eigene SDK wird benötigt), damit lässt sich zusätzlich Geld verdienen. Beim Vertrieb kassiert Microsoft 30% der Einnahmen aus dem Verkauf, eine Reduzierung auf 20% erfolgt, wenn die App einen Gesamtumsatz von mehr als 25.000 $ erwirtschaftet hat. Mit der Eröffnung des App Stores, der weltweit in verschiedenen Sprachen zur Verfügung steht, wird sich wohl in Kürze eine große Vielfalt von Apps ergeben.

Fazit

Metro Style Anwendungen sind nicht für jede Art von Businessanwendungen geeignet. WinRT wurde speziell für die Bedürfnisse von Metro Style geschaffen, so kann man etwa mit WinRT keine Desktop Anwendungen erstellen. Der Funktionsumfang der Klassenbibliothek ist gegenüber .net noch gering. Ein entscheidender Punkt für den Einsatz von WinRT ist auch, dass dies erst mit Windows 8 zur Verfügung steht und seitens Microsoft kein abwärtskompatibles WinRT Paket angeboten wird. Dafür wird ein Upgrade des Betriebssystems von Windows 7 auf Windows 8 günstig angeboten. Allerdings haben viele große Firmen erst vor kurzem von Windows XP auf Windows 7 upgedatet und planen wohl nicht schon wieder das nächste große Rollout. Nachdem von vielen Windows Vista übersprungen wurde, gilt es zu bedenken, dass es mit Windows 8 ebenso möglich ist. Wenn es gelingt, den Funktionsumfang von WinRT zu verbessern und Drittanbieter von WinRT Komponenten auf den Markt kommen, wird sich die Verbreitung von WinRT und Metro durchsetzen und irgendwann kein Weg mehr daran vorbei führen.

Windows 8-style UI - formaly known as *„Metro“*

Microsoft hat Windows 8, Server 2012 und Visual Studio 2012 seitens der Entwicklung in den *RTM (release to manufacture)* Status gebracht, damit gehen die Produkte bald ins DVD Presswerk und in die MSDN als Download. Blöderweise hat kurz nach der Ankündigung die *Metro AG* einen Einspruch bezüglich der Marke erhoben. Nun steht Microsoft ungünstig da, es wird zwar versucht den Konflikt herunterzuspielen, da es sich laut Microsoft nur um einen internen Codename handeln soll. Allerdings kann man das so auch nicht sehen, *Metro* ist zwar intern seit 2006 als Codename geführt, aber für das Smartphone bereits seit 2010 in dieser Form auch am Markt. Seither wurde strategisch unter dem Namen *Metro* sehr viel Marketing unternommen, um *Metro* auch bei neuen Microsoft Produkte mehr Bekanntheit zu verschaffen, eine sehr lange Zeit. Die Anwälte freuen sich wohl nun über den Geldzuwachs durch die Schlichtung der Markenrechte. Microsoft hat inzwischen reagiert und den Namen *Metro* vorübergehend als *Windows 8-style UI* ersetzt. Bleibt die Zeit vor dem Verkauf abzuwarten, wie der finale Name lauten wird.