Blockbuster

Microsoft Windows CE .net

Mit .Net scheint Microsoft ein guter Wurf gelungen zu sein. Zumindest Systemarchitektur und Entwicklungsumgebung dieser Basis-Technologie versprechen einiges – auch für den Automatisierungsmarkt. Nach dem großen Bruder Windows XP bringt Microsoft mit CE.Net nun auch die Plattform für den embedded-Bereich heraus. Mit geringen Lizenzkosten, harter Echtzeitfähigkeit, freiem Quellcode-Zugang und kleinem Speicherbedarf gepaart mit .Net-Kompatibilität hat CE .Net das Zeug zum Blockbuster in der Automatisierungsbranche.

Microsoft Windows CE .Net ist seit Mitte Januar allgemein verfügbar." Mit diesen Worten begann Scott Horn, Director Embedded and Appliance Platforms Group von Microsoft, seine Präsentation beim offiziellen Launch von Windows CE.Net in München. Mit dem Embedded-Betriebssystem hat Microsoft natürlich zunächst Massenmärkte wie Handheld-Computer, Smartphones, Set-Top-Boxen und mobile Kassenterminals im Visier. „Geräte, die mit Windows CE .Net laufen, verwirklichen unsere Vision, Informationen jederzeit, an jedem Ort und auf jedem Gerät verfügbar zu machen", so Horn.
Durch die harte Echtzeitfähigkeit und geringen Ressourcenbedarf wird es aber sicherlich wie schon seine Vorgänger auch für industrielle Geräte interessant – nicht nur fürs Bedienen und Beobachten, sondern auch für echte Steuerungskomponenten.

Robuste Plattform auch für die Automation

Mit CE .NET entwickelte Betriebssysteme können so optimiert werden, dass sie mit rund 200 KB Arbeitsspeicher auskommen. Das Betriebssystem unterstützt aktuelle Standards für die drahtlose Kommunikation wie Bluetooth und 802.11 und enthält die Technik von Microsoft Internet Explorer 5.5, Windows Media 8 und DirectX 8. Die Kompatibilität mit den Prozessortypen x86, Xscale, ARM, MIPS und der SH-Serie sorgt dafür, dass man freie Hand bei der Wahl der Hardwarearchitektur hat. Dies war bei CE 3.0 mitunter ein K.-o.-Kriterium. Verschiedene Board Support Packages und unterstützende Funktionen wie ein Platform Wizard sollen die Portierung von CE .Net auf Endgeräte erleichtern.

„Windows CE .Net ist die Embedded-Plattform der nächsten Generation." Scott Horn, Direktor Microsoft Embedded and Appliance Platforms Group

Mit Hilfe der kostenlosen .Net Emulation Edition (für Win 2000 und XP) lässt sich Software auch ohne das entsprechende Endgerät schreiben und testen. Eine Evaluation Edition ermöglicht es CE .Net an einem eigenen Embedded-Projekt zu testen. Parallel dazu gibt es ein praxisorientiertes Schulungsprogramm. In den ersten 90 Tagen ist die Entwicklungsumgebung von CE .Net zum Einführungspreis von 995 US-$ erhältlich. In der Entwicklungsumgebung sieht auch Heinrich Munz, Leiter Vorentwicklung von Kuka Roboter, einen entscheidenden Vorteil. Er stellte als strategischer Partner von Microsoft eine erste Implementierung von CE. Net und XP embedded vor (s. a. Interview).
Auch was den Preis anbelangt, ist man mit 15 $ bei Einzellizenzen in akzeptable Regionen vorgestoßen – zumindest für die Automatisierungsbranche, „zumal wir hiermit keinerlei funktionale Einschränkungen verbinden", betonte der Microsoft-Direktor.
Immer öfter wird gefordert, Endgeräte mittels intelligenter Anwendungen in die .Net-Infrastruktur einbinden zu können. Informationen jederzeit und an jedem Ort verfügbar zu haben, ist ein Kerngedanke der .Net-Vision. Damit dies möglich wird integriert Microsoft das .Net Compact Framework in CE .Net. Das gegenwärtig als Technology Preview verfügbare .Net Compact Framework ist eine Plattform zur Entwicklung von XML-Webservices und intelligenten Endgeräten.

Mit Mira wird das Webpad zum mobilen XP-Terminal

Es unterstützt mehrere Programmiersprachen, darunter Visual Basic .Net und Visual C# .Net, ermöglicht die Speicherverwaltung und Steuerung von Sicherheitsmechanismen und verbessert die Zuverlässigkeit der Anwendungen auf den Endgeräten. Mit den Smart Device Extensions for Visual Studio .Net werden die über vier Millionen Entwickler, die mit Visual Studio arbeiten, Anwendungen für CE .Net-Geräte mit den gleichen Programmierwerkzeugen schreiben können, die sie bereits für Desktop- und Serveranwendungen nutzen.
Die Betriebssysteme und Programmierwerkzeuge der Produktfamilie Windows Embedded bilden umfassende Softwareplattformen zur Ent-

Steckbrief Win CE.NET

- Echtzeitfähig (2,8µs Latenzzeit bei ISR, 17,9µs bei IST)
- Kleinster Speicherbedarf: 210kB
- Im Vergleich zu CE3.0 70% schnellerer Datenzugriff
- Doppelt so schnelle Netzwerkkommunikation
- Fünf Prozessortypen
- Einzellizenz (Vollversion): 15 $

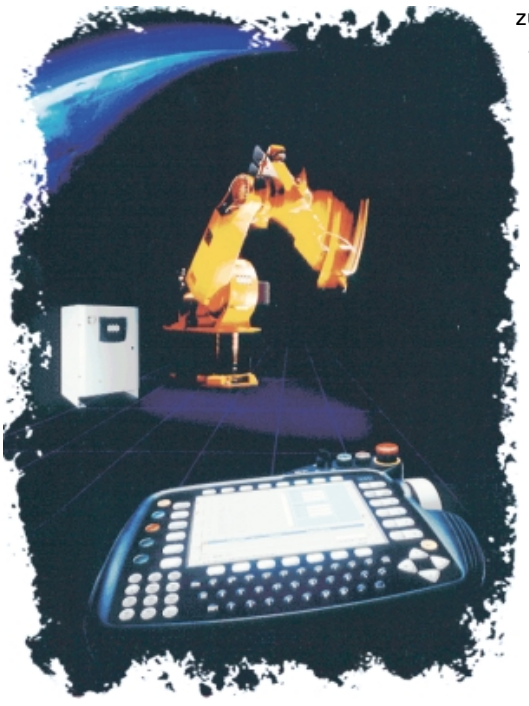
wicklung von Anwendungen für intelligente 32-Bit-Geräte der nächsten Generation. Derzeit gibt es die Embedded-Betriebssysteme CE .Net und XP Embedded. Unter dem Codenamen 'Mira' entwickelt Microsoft gegenwärtig eine Sammlung von Techniken auf Basis von Windows CE .Net, die Windows XP auf drahtlosen intelligenten Displays verfügbar macht.

Kuka ist als strategischer Partner eines der ersten Unternehmen, die CE.Net erprobt und als Plattform für ein mobiles Bedienkonzept implementiert haben. Über die Erfahrungen und strategische Bedeutung von .Net sprachen wir am Rande der Microsoft-Pressekonferenz mit Heinrich Munz, Leiter Vorentwicklung bei Kuka Roboter.

Was muss man sich unter .Net vorstellen, nur ein anderes oder auch ein besseres Konzept wie es SUN mit Java und der VM realisiert hat?
Es ist definitiv ein besseres Konzept, aber nichts wirklich Neues, sondern vielmehr der gelungene Versuch, die besten Ideen der IT-Vergangenheit zusammenzufassen und in einem kompatiblen und durchgängigen Konzept umzusetzen.

Welche Ideen sind das?
Sowohl Java als auch .Net haben die Grundidee, Computerprogramme maschinen- und betriebssystemunabhängig ausführen zu lassen. Dies ist die Grundvoraussetzung für über Netzwerke verteilte, leistungsfähige Applikationen. Das geht weit über das Surfen mit mehr oder weniger statischen Web-Seiten hinaus. So versteht sich denn auch die Aussage 'Programming the Web' ist nach 'Surfing the Web' die nächste IT-Welle, welche auf uns zurollen wird. Sun hat dies schon vor Jahren sehr treffend bezeichnet – „Das Netzwerk ist der Computer."

Was muss man sich darunter nun konkret vorstellen?
Gemeint ist damit, einem autorisierten Anwender potenziell die Gesamtheit aller vernetzten Computer zur Verfügung zu stellen. Dies steht im Gegensatz zur bisherigen zentralistischen Client-Server-Sichtweise, bei der ein Anwender genau wissen muss, wer der Client- und wer der Server-Computer ist und welche Funktionen er bereitstellt.
Um das zu erreichen, haben beide Konzepte ein Pseudo-Betriebssystem, das die Programme vom realen Betriebssystem abschottet. Sun nennt dies Virtual Machine; Microsoft .Net Framework. Die Programme sind also nicht direkt für das jeweilige Prozessor/Betriebssystem-Duo zugeschnitten, sondern werden zunächst in einem neutralen Zwischencode gespeichert. Bei Sun heißt das Byte-Code; bei Microsoft Common Intermediate Language (CIL) bzw. das Konzept Common Language Runtime (CLR). Erst auf dem Zielsystem und wenn gebraucht, wird dieser Code dann interpretiert (ursprünglich bei Java) oder kompiliert (grundsätzlich bei .Net) und ausgeführt.



Künftige Kuka-Roboter-steuerungen basieren auf .Net-fähigen Betriebssystemen von Microsoft

Bislang scheint alles alter Wein in neuen Schläuchen zu sein?
Hier zeigt sich der erste Unterschied. Suns Byte-Code ist für eine interpretierende Ausführung optimiert, während Microsoft auf einen kompilierenden Code setzt. Java Byte Code wird zwar mittlerweile auch von den so genannten Just in Time Compilern (JIT) in direkten Laufzeitcode übersetzt, dieser kann aber nie so optimal sein, wie ein generell zum Kompilieren vorgesehener Code.
Der wahrscheinlich gravierendste Unterschied ist jedoch, dass sich Byte Code eben nur in Java programmieren lässt. Dies zwingt jeden Entwickler von verteilten Netzapplikationen in Java zu programmieren. In Anbetracht der Millionen C/C++- und Visual Basic-Programmierer ein Killerkriterium – für Java. .Net lässt dagegen nahezu jede beliebige Programmiersprache zu – auch Java. Diese werden dann in den neutralen Zwischencode CIL übersetzt. Microsoft selbst bietet mit C/C++, C# und Visual Basic bereits einige Compiler für .Net an, weitere sind angekündigt. Zusätzlich unterstützt Microsoft andere Initiativen, die Compiler für Sprachen wie Oberon, Modula und Forth entwickeln.

Heinrich Munz, Leiter Vorentwicklung bei Kuka Roboter: „Bei der Masse an Vorteilen sehe ich einer rasanten Verbreitung von .Net nichts im Wege stehen."

Die CIL ist also eine Art Software-Ursuppe...?
...woraus man erstmalig Softwaremodule, die in unterschiedlichen Sprachen entwickelt wurden, problemlos zusammenlinken kann. Dies liegt hauptsächlich an der Tatsache, dass die CIL Datentypen, Aufrufkonventionen etc. genau definiert und vorgibt. Daran müssen sich alle Sprach-Compiler halten. Somit können problemlos interdisziplinäre Projekte entstehen, bei denen die Oberfläche z. B. in VB und die Logik in C# programmiert ist. Diese Möglichkeit, bestehendes Know-how weiter anwenden zu können, ist als Investitionsschutz ein weiterer großer Vorteil von .Net. ▸

Hinzu kommt, dass es bereits einige Open-Source-Aktivitäten gibt, das .Net Framework auch für andere Betriebssysteme wie Linux anzupassen. Microsoft selbst will .Net Frameworks für Konkurrenz-Betriebssysteme anbieten – freilich mit eingeschränkter Funktionalität.

Könnte man dann nicht auch SPS-Programmiersprachen unter die Suppe mischen?

Da habe ich meine Bedenken, da die wichtigste Voraussetzung für Steuerungsapplikationen – die Echtzeitfähigkeit – nach heutigem Kenntnisstand bei Einsatz von .Net Code unter Windows CE nicht gegeben ist. Das Problem liegt u. a. am so genannten Managed-Code. Der entsteht, wenn man mit .Net programmiert. Er lässt sich zwar verteilen und läuft auf jedem Gerät, ist aber zunächst nicht echtzeitfähig. CE-Applikationen sind aber nur dann deterministisch, wenn sie direkt auf das Betriebssystem zugeschnitten geschrieben wurden. Man spricht dabei von Native Code.

Beide Programmiermethoden sind zwar prinzipiell für CE.Net möglich, haben die genannten Vor- und Nachteile, bedingen aber zwei verschiedene Entwicklungsumgebungen. Während Managed Code mit Visual Studio.Net in den genannten Sprachen erstellt werden kann, braucht man für die Erzeugung von echtzeitfähigem native Code wie bisher das Tool Embedded Visual C++. Hier lässt sich allerdings nur in C oder C++ programmieren. Inwieweit hier Weiterentwicklungen Verbesserungen bringen muss sich erst noch zeigen. Windows CE war bis zur Version 3.0 auch nicht hart echtzeitfähig.

Es scheint also nur eine Frage der Zeit zu sein, bis es die ersten IEC-1131-3-Implementierungen für .Net gibt.

Dazu müsste Microsoft in der Architektur der Common-Language Runtime von .Net entsprechende Dinge ändern. Beispielsweise einen Subset von Managed-Code herausbringen, bei dem auf Echtzeit-Killer wie Garbage-Collection verzichtet wird. Vielleicht lässt sich das auch über Third Parties realisieren. Nur war bisher einfach noch zu wenig Zeit, um dies genau zu untersuchen. Jetzt mit dem offiziellen Release kann man diese ganzen Dinge wirklich auf Herz und Nieren prüfen. Natürlich haben wir auch schon darüber nachgedacht, unsere Roboter-Programmiersprache als Common-Language-Runtime auszuprägen und sie mit Visual Studio .Net eben runterzuschreiben, den Compiler-Knopf zu drücken und damit in diese Zwischensprache zu übersetzen. Aber da gibt es noch einige Hürden zu nehmen.

Welche Vorteile hat .Net für den Anwender

Der Entwickler profitiert in der beschriebenen Form, ergänzt um hocheffiziente Entwicklungswerkzeuge.

„Um echtzeitfähige Programme in .Net, z. B. für eine SPS, schreiben zu können, sind Ergänzungen an der Common Language Runtime notwendig"

Der Endanwender wird eine völlig neue Generation von Netzwerkdiensten und Geräteunabhängigkeit 'erfahren'. Dank .Net wird es möglich sein, dass der Anwender seine Arbeiten mit ein- und demselben gekauften oder gemieteten Programm vom PC, Laptop, PDA, Fernseher, Web-Pad, Spielekonsole, Handy, Auto-Bordcomputer usw. durchführen kann. Dies ist ein weiterer Unterschied der neuen .Net Generation zu Java. Während Java vorwiegend auf Computern zum Einsatz kam, wird .Net Einzug in alle mögliche Arten von Geräten finden, den Embedded Devices. Dies ist auch der Grund, weshalb Microsoft so intensiv die Weiterentwicklung von Windows CE betreibt. Nicht von ungefähr heißt die neueste Version nicht CE 4.0, sondern CE .Net.

Wie sieht die Umsetzung von .Net bei Kuka aus?

Zunächst einmal wird Kuka Windows CE.Net in das Bediengerät des Roboters übernehmen und dessen PC-basierte Steuerung mit Windows XP Embedded ausstatten – wie bisher inklusive unserer eigenen Echtzeiterweiterung, denn XP Embedded ist wie NT Embedded nicht von Haus aus echtzeitfähig. Diese Technologie bieten wir übrigens stand alone am freien Markt an. Jeder Steuerungshersteller kann damit von unserer Erfahrung profitieren.

Die Verbindungen zwischen Bediengerät und Robotern aber auch zwischen den Robotern untereinander und zu anderen Automatisierungsgeräten wird über TCP/IP-Ethernet erfolgen. Dadurch haben wir die ideale Infrastruktur für das .Net-Umfeld geschaffen. Welche Dienste wir hierauf in Zukunft anbieten und somit Kundenmehrwert schaffen, kann ich an dieser Stelle aus verständlichen Gründen nicht näher ausführen.

Zur Vernetzung von Steuerungen braucht man ein echtzeitfähiges Ethernet. Wie sieht es denn damit aus?

Das Real-Time-Transport-Protokoll von .Net ist hier ein Schritt in die richtige Richtung. Es unterstützt über die Priorisierung von Telegrammen die Echtzeitfähigkeit von Ethernet. Sollte dann viel Traffic auf dem Netz sein, haben die wichtigen Steuerungsmeldungen Vorrang. Dafür gibt es sogar entsprechende IEEE-Normen, nur bisher kenne ich keinen Betriebssystemhersteller außer Microsoft, der sie wirklich schon in seine TCP/IP-Stacks implementiert hat. Das ist für uns ebenfalls ein ganz wichtiger Aspekt.

Bietet Kuka nur die Echtzeiterweiterung für XP an, oder künftig auch Kuka-Steuerungstechnik?

OEM-Kunden können von uns alle Steuerungskomponenten beziehen, die wir selbst für unsere Roboter einsetzen. Dies beginnt bei der PC- oder Panel-Hardware, geht über die Echtzeiterweiterungen und Anschaltungen für diverse Feldbusse bis hin zu Steuerungsprogrammen, wie Soft-SPS, HMI-Studio, OPC-Server. Selbstverständlich ist es auch möglich, eine komplette Steuerung z. B. zum Betrieb von Portalrobotern oder anderen Kinematiken zu bekommen.

Wann fließen Ihre Erfahrungen in die Serie ein?

Um es klar zu sagen. Die Implementierung ist zur Zeit ein absoluter Prototyp, mit dem wir selbst erst einmal Erfahrungen sammeln und mögliche Fehlerquellen und Probleme identifizieren wollen – das machen wir nicht wie manch anderer beim Kunden, sondern im Labor. Erst wenn wir von der Einsatzreife absolut überzeugt sind und unsere Kunden das Konzept mittragen, werden wir damit in Serie gehen. □