Middleware – die Zukunft für Test und Automatisierung?

Dipl.-Ing. Robert Schachner, RST Industrie Automation

Das Thema „Industrie 4.0“ ist als Zukunftsvision der verarbeitenden Industrie in aller Munde. Obwohl – oder gerade weil – die Definitionen schwammig bleiben und viele Firmen versuchen, ihre eigene „Standards“ zu platzieren, ist es umso wichtiger, sich jetzt mit den damit einhergehenden Herausforderungen auseinanderzusetzen und Lösungen zu suchen.

Auch der Markt für Testsysteme ist in ständiger Bewegung. Vorbei sind die Zeiten, in denen Standalone-Systeme einfache Testaufgaben ausgeführt haben. Heutzutage spricht man nicht mehr nur von HIL (Hardware in the Loop) sondern auch von MIL und SIL (Model in the Loop und Software in the Loop). Darüber hinaus tritt auch der Integrationstest immer mehr in den Vordergrund. Hier testen Systeme das Zusammenspiel der einzelnen Komponenten untereinander. Um diesen neuen Vorgehensweisen und Herausforderungen begegnen zu können, sind modulare und offene Testtechnologien nötig

Der Einsatz moderner Middlewareplattformen kann helfen, den neuen Herausforderungen in beiden Branchen zu begegnen.

Middleware – ein Überblick

Die zentrale Funktion von Middlewareplattformen ist die Kommunikation. Prozesse müssen Daten austauschen, sich untereinander koordinieren und I/O Daten verarbeiten. Grundsätzlich können wir zu dieser Grundaufgabe drei Ausgestaltungen unterscheiden: das klassische Prozessdatenmodell, samplebasierte Datenströme und messagebasierte Kommunikation. Im Folgenden wollen wir kurz die Prinzipien und Funktionsweisen, sowie Vor- und Nachteile der drei primären Kommunikationsformen beleuchten.

Das klassische Prozessdatenmodell ist eine zentrale Struktur, in der stets nur die aktuellsten Daten vorliegen. Ein hierarchisch strukturiertes Datenmodell liefert ein reales Abbild, das sich zeitlich so nah wie möglich an den mechanischen Abläufen orientiert. Neuere Daten lösen dabei durch überschreiben ältere Daten ab. Prozesse können so sehr schnell aktuelle Daten lesen und Schreiben. Es ist daher ideal für zeitdiskrete Echtzeit Anwendungen wie zum Beispiel bei Regelungsabläufen. Dieses Verfahren ist auch typisch für klassische SPS-Steuerungen.

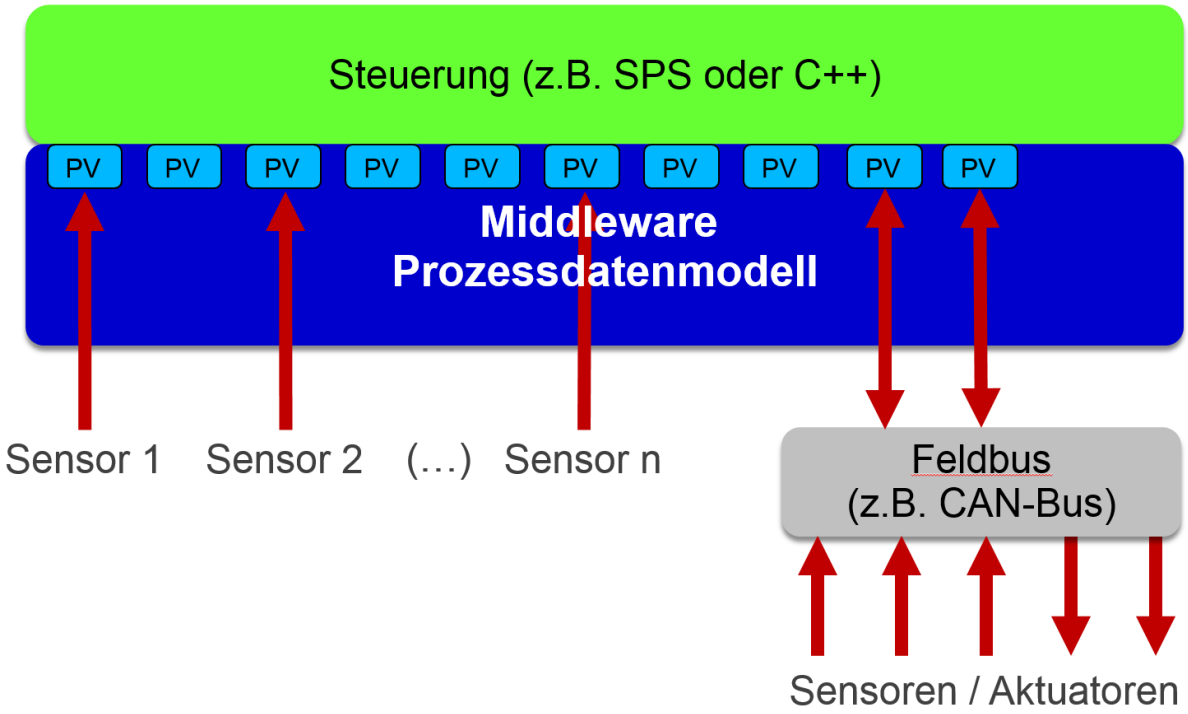

Abb. 1: Kommunikation im klassischen Prozessdatenmodell

Vorteile:

- Logische Anweisungen können sehr schnell ausgeführt werden.
- Zeitdiskrete Abläufe, wie zum Beispiel Regler, lassen sich hervorragend ausführen.

Nachteile:

- Es wird immer nur der neueste Wert gespeichert, ältere Daten gehen jedoch verloren. So kann es bereits bei Kommandos oder Fehlernummern über das Prozessdatenmodell leicht zu Inkonsistenzen kommen (Im SPS-Bereich wird dies toleriert).

Samplebasierte Datenströme (oder FIFOs) sind direkte Punkt-zu-Punkt-Verbindungen zwischen zwei Prozessen, wobei die Rollen von Sender und Empfänger von Anfang an festgelegt sind. Der sendende Prozess schreibt seine Daten in einen Puffer, der vom empfangenden Prozess ausgelesen wird. Diese Vorgehensweise ist ideal für asymmetrische Kommunikationsvorgänge und umfangreiche Datenströme. Ein Einsatzfeld dafür sind zum Beispiel schnelles Analog Sampling oder Audiostreams.

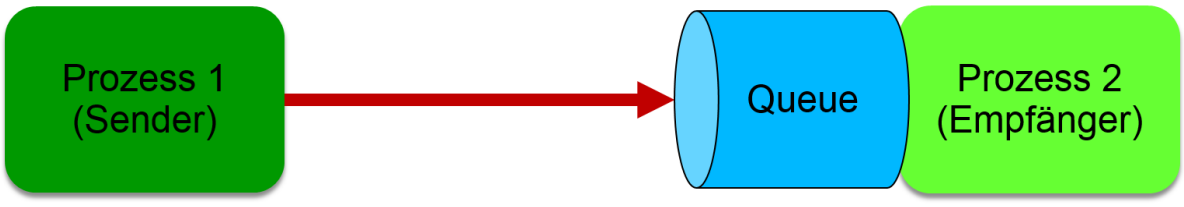

Abb. 2: Kommunikation über samplebasierte Datenströme

Vorteile:

- Die Verarbeitungsgeschwindigkeit ist ähnlich der bereits beschriebenen Prozessvariablen.
- Die Verarbeitung kann asynchron zum Ereignis erfolgen.
- Daten gehen durch FIFO-Pufferung nicht verloren.

Nachteile:

- Die Daten können nur für einen Prozess bereitgestellt werden (1:1).
- Die zeitliche Zuordnung erfolgt nur indirekt über einen Timestamp.

Die messagebasierte Kommunikation verteilt über einen Message Broker serialisierte Daten von und zu den einzelnen Prozessen. Dies geschieht entweder als Punkt-zu-Punkt-Verbindung oder über den Publish / Subscribe-Mechanismus, in dem Sender (Publisher) Daten über Informationskanäle (Topics) veröffentlichen und Empfänger (Subscriber) die für sie jeweils relevanten Themen abonnieren. Da Sender und Empfänger für den Datenaustausch nicht mehr direkt verbunden sind, müssen die Datenströme wischen Systemen wie zum Beispiel bei Feldbussen nicht mehr manuell „verdrahtet“ werden. Anlagenteile einer Maschine können sich so ohne Konfiguration zu einem Gesamtsystem zusammenfügen.
Mit Hilfe von Remote Procedure Calls können Funktionen außerhalb des lokalen Prozesskontexts ausgeführt werden. Dies ermöglicht den Aufbau von individuellen Funktionsbibliotheken(z.B. Alarm Management), die systemübergreifend ausgeführt werden.

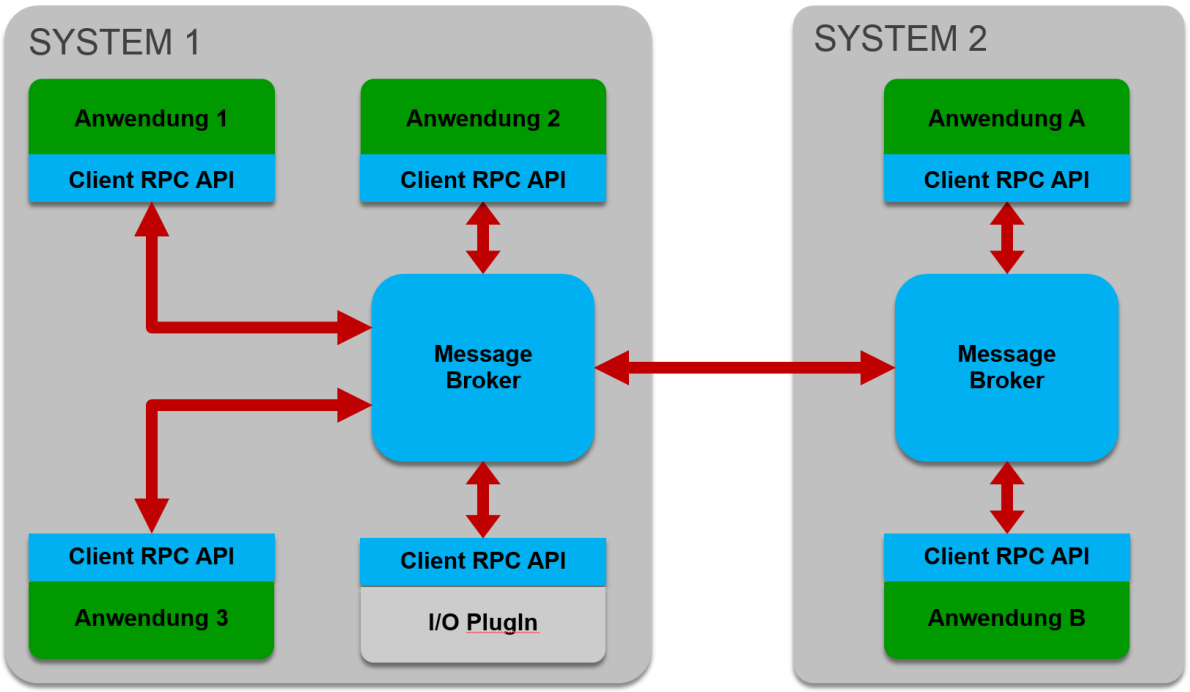

Abb. 3: Komplexe Kommunikation über Message Broker

Vorteile:

- Datenverbindungen können 1:1 oder n:m ausgestaltet sein.
- Durch die Implementierung als Push-Dienst verpassen Prozesse keine Wertänderungen.

Nachteile:

- Gerade bei größeren Datenmengen geht die zeitliche Zuordnung verloren.
- Serialisierung und Brokerage führen zu höheren Latenzzeiten.

Aber auch über die Kommunikation und Datenverwaltung hinaus bieten Middleware Technologien zahlreiche weitere Funktionen, um Applikationsentwickler und Systemintegratoren zu unterstützen. Hier einige Beispiele aus der Gamma V Middlewareplattform.

Zur zeitlichen Synchronisation und Kontrolle von Prozessen stellt die Plattform ein Zeitmodell bereit:

- **Loops** aktivieren Prozesse und Funktionen der Middleware

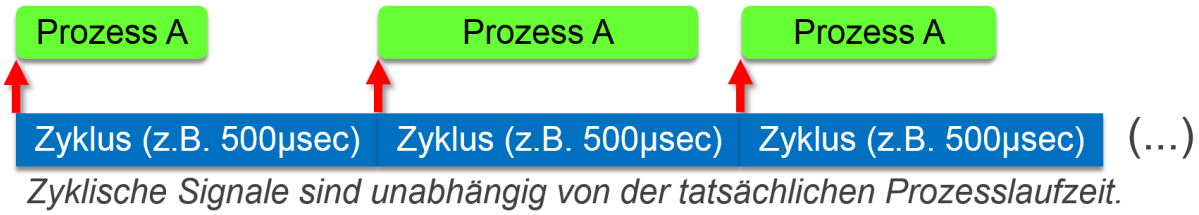

Abb. 4: Loop als Trigger für zyklische Aktionen

- **Ein Scheduler** koordiniert die Ausführung von Prozessen und Middleware Funktionen auch im Multicore Betrieb. Durch die Definition von Abhängigkeiten kann so die zeitliche Abfolge aller relevanten Funktionen und Programme koordiniert werden.

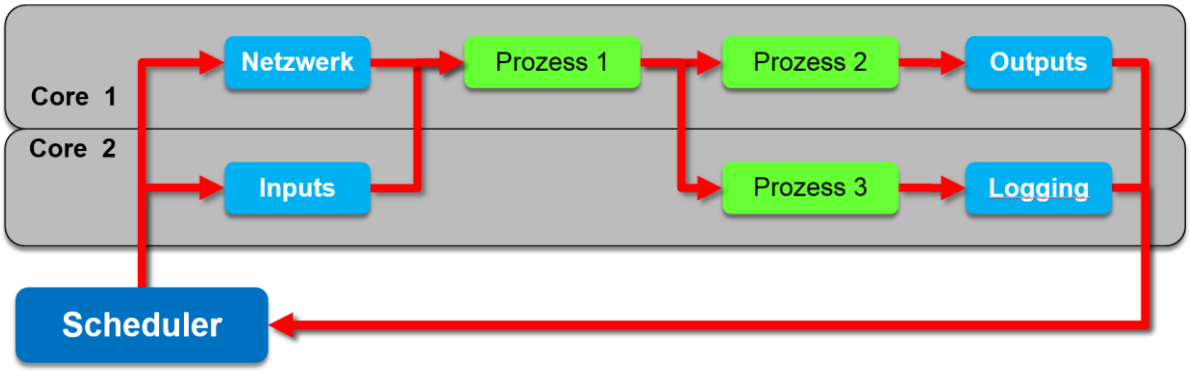

Abb. 5: Ein einfacher Ablauf im Scheduler

- **Über Actions oder Hardware-Interrupts** können Prozesse und Funktionen durch Ereignisse an der I/O aktiviert werden.

Außerdem stellt das klassische Gamma-Prozessdatenmodell zahlreiche Funktionen zur Verfügung, die für jede Applikation genutzt werden können, ohne jedesmal den dafür nötigen Programmieraufwand zu betreiben:

- **Polling / Signal On Change:** Prozesse pollen Ihre Variablen oder lassen sich wecken, wenn Änderungen vorliegen.
- **Atomare Locks:** Zur Synchronisation von Zugriffen durch unterschiedliche Prozesse.
- **I/O Mapping:** Hardwarekanäle können direkt auf Prozessvariablen abgebildet werden. Auslesen und Setzen von I/Os erfolgt transparent durch Lese- und Schreibzugriffe auf die verknüpfte Variable.
- **Zugriffsrechte:** Schreib- und Leserechte können für einzelne Variablen oder Variablengruppen gesetzt werden.
- **Limit Checking:** Bei Schreibzugriffen auf Variablen werden einstellbare Grenzwerte überwacht.
- **Logging:** Daten können zur Laufzeit mit einem Timestamp gelogged werden.
- **Simulation:** Für Test- und Simulation von Maschinen und Prüflingen bietet die Plattform eine extrem vielseitige Simulationsfunktion.

Darüber hinaus kann das Datenmodell zur Laufzeit dynamisch erweitert werden.

Die Integration von I/O-Hardware erfolgt über Plug-Ins. Das erleichtert Hardwareupgrades und Change Management. Durch das Vorhandensein einer offenen Schnittstelle ist es problemlos möglich, für eigene Hardware schnell und einfach Plug-Ins zu entwickeln.

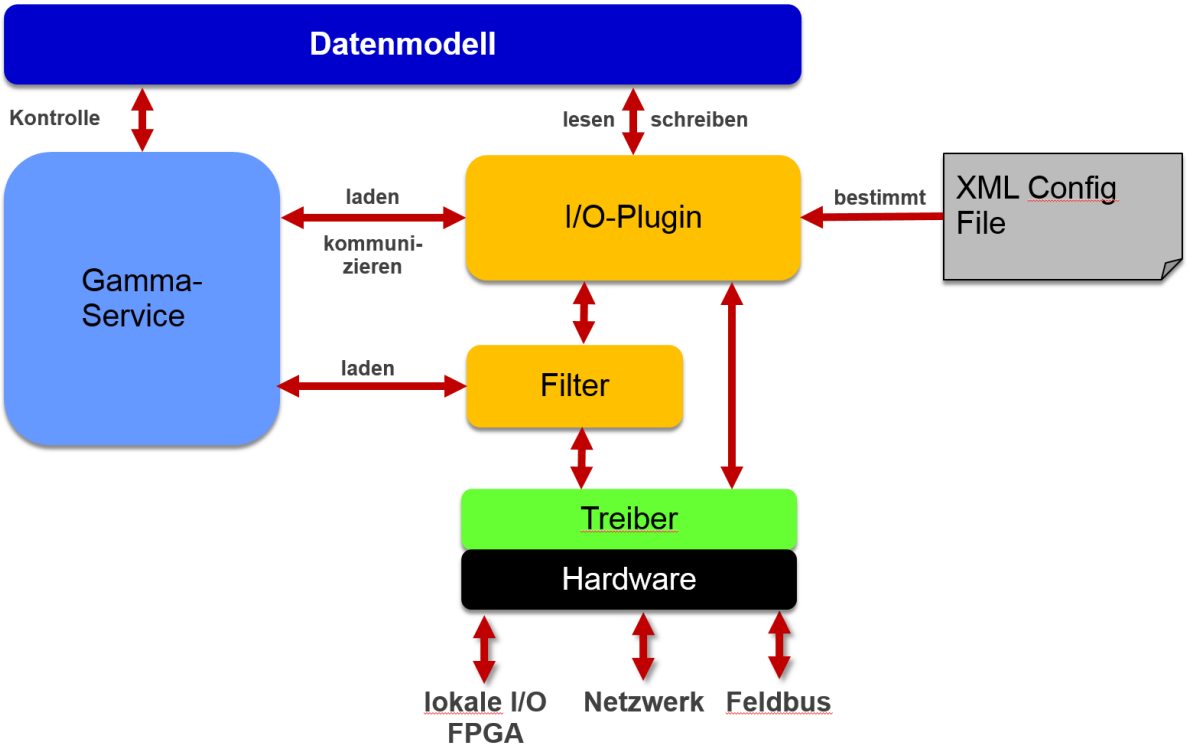

Abb. 6: I/O-Integration über Plug-Ins

Neben den in der Gamma Middleware bereits zahlreich vorhandenen I/O PlugIns lassen sich eigene Funktionalitäten sehr einfach integrieren. Sei es eine klassische I/O Karte oder ein Feldbus. Alles was Daten produziert, lässt sich integrieren.

Vernetzung und verteilte Systeme
Neben den Feldbussen, die über die I/O- bzw. Feldbusebene integriert werden, ermöglicht die Middleware auch eine Kommunikation über die Prozessebene. Damit ist die Middleware ideal für die Integration von Teilsystemen zu einem Gesamtsystem geeignet. Folgende Szenarien sind dabei möglich:

- Vernetzung des Prozessdatenmodells über Systemgrenzen hinweg.
- Integration weiterer systemübergreifender Dienste wie zum Beispiel Alarm Management, Benutzerverwaltung, historische Daten, Integration von Datenbanken
- Kommunikation mit Android basierten Apps
- Ausweitung der Kommunikation auf Anlagenebene über OPC UA

Unsere Strategie für die Zukunft
Trotz Handlungsbedarf gibt es aktuell viele Fragen und wenig Antworten.
Unsere Strategie ist klar:

1. Mit den bereits heute verfügbaren Methoden und Werkzeugen entwickeln wir Testsysteme und Maschinen, die auch für zukünftige Entwicklungen gerüstet sind.
2. Durch die konsequente Verwendung offener Schnittstellen können neue Technologien zu jeder Zeit schnell und einfach adaptiert werden. Somit können wir auf neue Herausforderungen schnell und flexibel reagieren.

3. Die Integration dieser modularen Technologien braucht auch viele Köpfe. Deshalb engagieren wir uns in der Unternehmensvereinigung **Embedded4You e.V.**

Im Rahmen der langjährigen Zusammenarbeit mit dem Verein sind wir auf dem Weg schon weit vorangeschritten:

- Middleware als Kerntechnologie für die Vernetzung
- Werkzeuge, die je nach Anforderung an die Middleware angekoppelt / kombiniert werden können:
    - SPS-Werkzeuge: IEC61131-3 oder IEC61499
    - Visualisierungswerkzeuge: XIBASE oder QT
    - IT-basierte Werkzeuge: C/C++, Python, eTrice
    - Testwerkzeuge: IsyTester, ITE, CCDL, MBTSuite
    - Simulation (Basierend auf Matlab Simulink)
- Hardware:
    - Industrielle Hardware oder günstige Small Computing Lösungen ähnlich Raspberry Pi oder Beaglebone Black
    - Embrick als neue Form der Modularität von I/O-Funktionen, die Kostenreduktionen bis zu 30% gegenüber Standardtechnologien ermöglicht

Gemeinsam sind wir gefordert, die Anforderungen der Zukunft bereits jetzt umzusetzen.

Autor
Dipl.-Ing. (FH) Robert Schachner bewegt sich im embedded-Markt seit 1985. Mit über 200 Projekten basierend auf Middleware Technologien schöpft er aus einem großen Potential an Erfahrungen.
1993 gründete er die RST Industrie Automation GmbH mit der er seither erfolgreich Middleware Plattformen entwickelt und in Projekten einsetzt. Herr Schachner ist Gründungsmitglied und zweiter Vorstand im Verein Embedded4You. Als Dozent der Hochschule Rosenheim lehrt er Kommunikationstechnologien.