Die LBS Programmbibliothek - erstes Open Source Modell **in** der Landwirtschaft

ACHIM SPANGLER, TUM-WEIHENSTEPHAN
HERMANN AUERNHAMMER, TUM-WEIHENSTEPHAN

Abstract

The open communication system "Landwirtschaftliches BUS System" (LBS) lacks a standard reJerence implementation, which would lead to capable and compatible communications systems. An Open Source program library could be developed to a standard implementation oJ the application layer oJ DIN9684. This would reduce the needed investments for developing LBS systems, and would guarantee compatibility 01 all systems by using the same source code.

1 Einleitung in das elektronische Kommunikationssystem LBS

Das Landwirtschaftliche BUS System (LBS) ist ein offenes Kommunikationssystem zur Steuerung und Überwachung von Arbeitsprozessen. Hierbei werden die Jobrechner der Arbeitsgeräte und des Traktors über CAN untereinander und mit Diensten wie z.B. virtuellem Termial und Task-Controller verbunden.

Bei LBS wird ein Arbeitsgespann als Netz weitgehend autonomer Einheiten modelliert. Diese bieten nach außen Dienste an, und müssen zum Teil selbst die Dienste anderer Elemente in Anspruch nehmen. Zur von Konstruktionsdetails abstrahierten Steuerung und Überwachung von Arbeiten werden hierbei Prozessdaten verwendet, mit denen eine Einheit ähnlich zu "Common Object Request Broker" (CORBA) durch die Sammlung seiner Dienstleistungsobjekte im Netz dargestellt werden kann. Die Zugriffe zu den Diensten eines Prozessdatenobjektes sind für alle Prozessgrößen gleich.

Nur sehr wenige Informationen werden in festen Zeitintervallen, ohne direkte Anforderung gesandt. Die als Basisdaten definierten Informationen stellen ein wichtige Grundlage zur Arbeit aller Systemteilnehmer dar. Theoretische und wahre Geschwindigkeit, Zapfwellen- und Motordrehzahlen geben den Anbaugeräten wichtige Auskünfte über das zentrale Antriebsgerät, den Traktor.

Die Benutzerstation und seine Möglichkeit zur Verwaltung von virtuellen Terminals kann durch .windows on a tractor" beschrieben werden. In der Art von Windows Anwendungen kann ein LBS Jobrechner die Dienste der Benutzerstation verwenden, um sich dem Bediener grafisch darzustellen und um seine Steuerung vorzunehmen.

2 Problemstellung aus der Analyse der Umsetzung der Norm durch Hersteller

Auch 3 Jahre nach Veröffentlichung der LBS Norm gibt es keine breite Palette von Geräten mit leistungsfähiger und normkonformer Nutzung des Protokolls. Die momentan am Markt verfügbaren Systeme setzen LBS zumeist in sehr unterschiedlichen Interpretationen um, was dem Landwirt große Probleme bereitet, wenn er versucht LBS Geräte verschiedener Hersteller zu kombinieren.

2.1 *Fehlen einheitlicher Implementierung*

Die meisten Landmaschinenhersteller haben ihre LBS Systeme ohne Abstimmung miteinander entwickelt, so dass bei nicht exakt definierten Elementen der Norm jeweils unterschiedliche Auslegungen gewählt wurden. Zudem wurden Kommentare veröffentlicht, die nicht ausreichend mit den entsprechenden Normpassagen abgeglichen waren.

Da sich bei den von den Landmaschinenherstellern eingesetzten Jobcontrollern der Mikroprozessor und das Betriebssystem bzw. BIOS stark voneinander unterscheiden, hätte eine zentral erstellte Referenzimplementierung mit einem hohen Verwaltungsaufwand an alle Systeme angepasst werden müssen. Ein kommerzieller Softwarehersteller hätte sich diesen Aufwand durch zu hohe Lizenzgebühren finanzieren lassen müssen, oder er hätte den Programmtext den Finnen offenlegen müssen, damit diese eigenständig eine Anpassung an ihr System vornehmen könnten.

2.2 Beschränkte Nutzung der Möglichkeiten des Kommunikationssystems

Die Komplexität der Prozesse in dem offenen Kommunikationssystem LBS hebt sich stark von den üblichen Abläufen in einem Jobcontroller einer landwirtschaftlichen Maschine ab. Daher wird das Leistungspotential von Prozessdaten und virtuellem Terminal mit den üblichen Programmiertechniken nur sehr unzureichend umgesetzt. Statt durch Investitionen und Flexibilität die benötigten Programmierweisen langfristig in die eigene Entwicklung einzubringen, werden Teillösungen gesucht, deren Leistungsgrenzen schon bei etwas komplexeren realen Bedingungen überschritten werden.

3 Lösungsansätze für eine standardkonforme LBS Umsetzung

Damit LBS eine größere Marktdurchdringung erfahren kann, müssen standardkonforme leistungsfähige Systeme verfügbar sein, die vom Kunden flexibel kombiniert eingesetzt werden können.

3.1 Einsatz einer Referenzimplementierung durch Open Source fördern

Unter Federführung einer neutralen Institution sollte in Zusammenarbeit mit Normentwicklern eine Referenzimplementierung entwickelt werden. Wird diese von Landmaschinenherstellern als Open Source Projekt gefördert, kann jeder Anwender den verfügbaren Quelltext an das eigene System anpassen. Die Aufgaben zum Test des Systems, zur Weiterentwicklung und zur Anpassung an diverse Jobrechner, können in einem Open Source Projekt auf alle Beteiligte verteilt werden, sofern jeder erkennt, dass der eigene Nutzen die eigene Investition übersteigt. Der Kunde würde dadurch Geräte frei kombinieren können, deren Kommunikationssystem einen leistungsfähigen, identischen Application Layer zu LBS verwenden.

3.2 An Teilaufgaben eines LBS-Systems orientierte modularisierte Programmbibliothek

Die einzelnen Aktivitäten eines LBS Systems lassen sich am besten durch die Verknüpfung von mehreren Teilaufgaben beschreiben. Jedes dieser Module kann so konzipiert werden, dass es die in seiner Schnittstelle angebotenen Dienste effizient und sicher auch in schwierigen Betriebsbedingungen umsetzt, und damit zur Umsetzung vieler Aufgaben optimal eingesetzt werden kann. Als Raster zur Modularisierung bietet sich die Unterteilung der LBS Norm in Bereiche wie Systemverwaltung, Prozessdaten und Benutzerstation an.
Durch die Modularisierung können für jedes Element des Kommunikationssystems Anwendungsszenarios entwickelt werden, um die Anforderungen an den jeweiligen Bereich der Software festlegen zu können. Zudem können die Module so ausgelegt werden, dass sie einen sicheren und einfachen Zugriff auf komplexe Elemente des Kommunikationssystems ermöglichen.

4 Entwicklung einer Open Source LBS Programmbibliothek

Im Rahmen der Forschergruppe "IKB-Dürnast" wurde eine LBS Programmbibliothek entwickelt. Diese Software wird zur Förderung von LBS als Open Source weiterentwickelt.

4.1 Pflichtenheft

Die Entwicklung eines hardwareunabhängigen Application Layer für ein LBS Kommunikationssystem, bei dem leistungsfähige Schnittstellen einen einfachen und sicheren Zugriff auf alle LBS Dienste für alle Plattformen in identischer Weise bereitstellen, kann als oberste Maxime der Projektentwicklung angesehen werden. Die Softwarestruktur muss die Integration von Erweiterungen, wie z.B. Strategien zur Behandlung von Interessenkonflikten und die optionale partielle ISO 11783 Kommunikation, ermöglichen, Daneben muss das Design den Anforderungen eines Echtzeitsystems entsprechen und sollte von Compilern für die üblichen Mikroprozessoren unterstützt werden. Zuletzt sollte die Programmbibliothek von den Entwicklern der Branche mit vertretbarem Einarbeitungsaufwand verstanden werden und sollte in bestehende Projekten integriert werden können.

*4.2 Analyse der Abläufe **in** einem LBS System*

Aufbauend auf einer Analyse der LBS Norm wurden Anwendungsszenarios entwickelt, um die Anforderungen an die Flexibilität und Leistungsfähigkeit der Implementierung abschätzen zu können. Daneben wurden Interpretationsspielräume und Mängel im Normtext ausgelotet und in Zusammenarbeit mit den Nonnentwicklem geklärt.

4.3 Aufbau der LBS-Lib

Zur Unterstützung der einfachen Portierbarkeit auf unterschiedliche Systeme wurde das System in einen komplett hardwareunabhängigen Application Layer und in eine hardwarenahe Schicht zum einheitlichen Zugriff auf die Hardware unterteilt. Abbildung 1 zeigt, wie die Teilaufgaben des LBS Kommunikationssystems weiter in Bereiche wie Systemverwaltung (LBS_System) und Prozessdaten (LBS_Process) aufgeteilt worden ist.

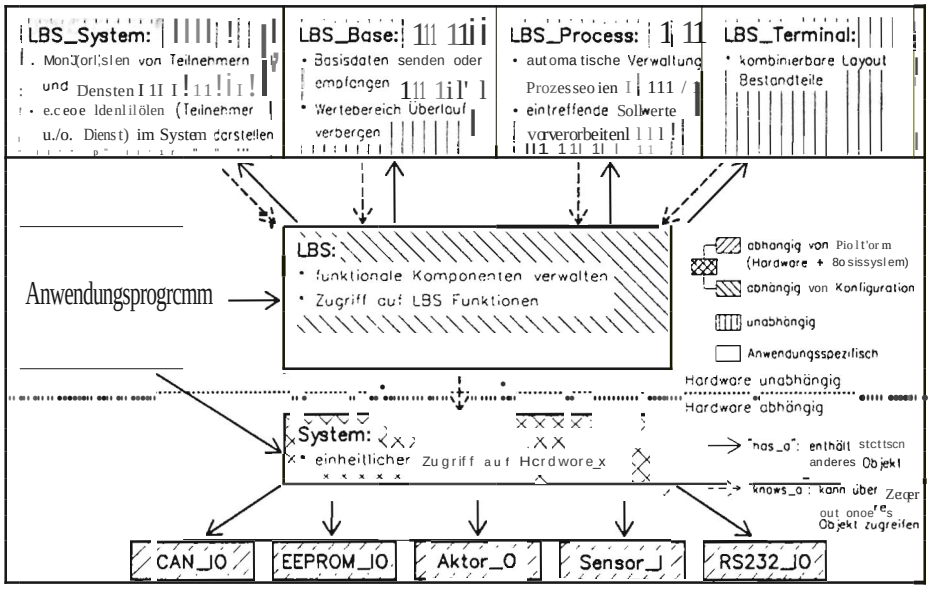

Abbildung 1: Darstellung der LBS Lib **im** Vollausbau

Für eine leistungsfähige und systemübergreifend einheitliche Nutzung von Hardware Resourcen wurden Module vorgesehen, die zum Teil nur bei Bedarf in ein Projekt eingefügt werden können (z.B. RS232_IO nur bei Einsatz der seriellen Schnittstelle). Diese ermöglichen auch die Entwicklung hardwareunabhängiger Anwendungen, so dass Beispielprogramme und ein Open Source Task-Controller möglich sind.

4.4 Implementierung

Die LBS Programmbibliothek wurde mit C++ implementiert, da es auf dem weit verbreiteten C aufbaut, und damit in den Bereichen Lernaufwand für Programmierer, Unterstützung durch Compiler und Integration in bestehende Projekte eindeutige Vorteile aufweist. Zudem unterstützt C++ als objektorientierte (OO) Programmiersprache die Implementierung der Module mit Objekten und deren Unterscheidung von Schnittstelle und Implementierungsdetails. Im Vergleich zu anderen OO Sprachen kann man bei C++ sehr gut den Overhead (z.B. vergrößerte Laufzeit) steuern, indem für einzelne Teilaufgaben eine Kosten-Nutzen Analyse der verfügbaren Implementierungsalternativen durchgefiihrt wird.

Die STL Container von C++ unterstützen die geforderte Flexibilität des Systems. Da die für Echtzeitsystem problematische Allozierung Speicher auf nicht zeitkritische Bereiche beschränkt werden kann, waren diese zumeist statischen Arrays vorzuziehen. Zur Steigerung der Effizienz von kurzen, zumeist dem Setzen und Auslesen von einfachen Werten dienenden Funktionen konnten die inline Funktionen von C++ eingesetzt werden.

5 Einsatz der LBS-Lib

Die LBS-Lib unterstützt die einfache und schnelle Entwicklung von leistungsfähigen Anwendungen wesentlich, indem komplexe Aufgaben durch einfache Funktionen zugänglich gemacht werden, und Routineabläufe automatisch ausgeführt werden. So reicht ein einziger Funktionsaufruf, um einen Teilnehmer am LBS anzumelden, für den bis zum Systemstop automatisch Aufgaben wie Senden einer regelmäßigen Alive Botschaft, oder Antworten auf Anfragen nach dem Bezeichner erfüllt werden. Entsprechend werden Messwertanfragen zu lokalen Prozessdaten automatisch beantwortet, und eintreffende Sollwerte werden für eine effiziente Entscheidung über Akzeptanz oder Ablehnung passende vorverarbeitet.

Die LBS-Lib wird mit einem simulierenden BIaS auf einem pe, auf realen Jobrechnern in einem Testlabor, und im Einsatz zur Datenerfassung bei landwirtschaftlichen Arbeiten getestet. Somit kann die Funktion unter simulierten Extrembedingungen und im Dauereinsatz untersucht werden.

6 LBS-Lib als Open Source Projekt

Zur Weiterentwicklung der LBS-Lib sollen die In Abbildung 2 dargestellten Kreisläufe

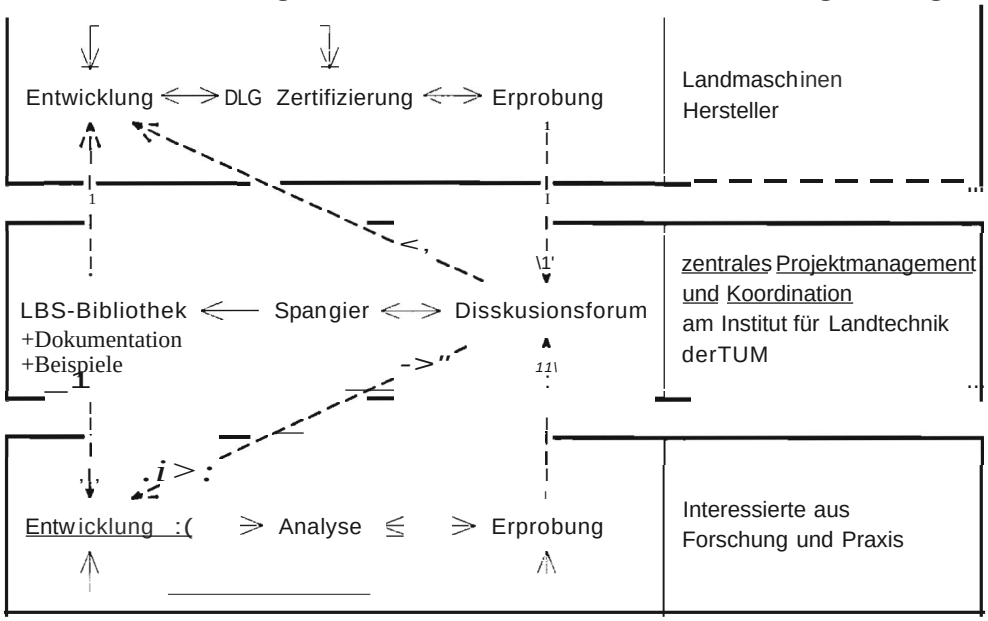

Abbildung 2: Kreisläufe der Entwicklung der Open-Source LBS-Lib

etabliert werden. Erfahrungsaustausch und Angebot der Quelltexte mit Dokumentation und Beispielen werden von zentraler Stelle im Institut für Landtechnik der TUM (<http://ikb.weihenstephan.de/deu/research/tp_2/actions.html» erfolgen. Im Verbund mit den anderen Bereichen von Wissenschaft, allgemeinen Anwendern und Interessierten und der Landmaschinenindustrie sollen ausführliche Tests, Portierung auf unterschiedliche Systeme und Erweiterung der Software gemeinsam betrieben werden.

7 Ausblick

Die LBS-Lib wird auf einem deutschen und englischen Workshop vorgestellt. Dort können alle Beteiligten auch Erfahrungen im Einsatz der Programmbibliothek zur Anwendungsentwicklung sammeln. Zudem wird dieses Open Source Projekt im November 2000 der japanischen Landmaschinenindustrie auf einem LBS Symposium vorgestellt. Wenn ausreichend viele Hersteller den Bedarf einer leistungsfähigen LBS Kommunikation erkennen, und feststellen, dass die LBS-Lib ein passendes Werkzeug dafür ist, könnte sich dieses Projekt langfristig auch über Deutschland hinaus zu einem de-facto Standard entwickeln.

8 Literatur

AUERNHAMMER, H.; DEMMEL, M. (2000): Innovationen in Technik und Bauwesen für eine wettbewerbsfähige und nachhaltige Landwirtschaft. In: KTBL-Schrift

AUERNHAMMER, H.; DEMMEL, M.; SPANGLER, A. (1999): Betriebsdatendokumentation mit LBS und GPS für Traktor-Gerätekombinationen. In: Tagung Landtechnik 1999: VDI Verlag, VDI Berichte 1503: 217-221

BALZERT, H. (1996): Lehrbuch der Software-Technik, Software-Entwicklung. Heidelberg, BerIin, Oxford: Spektrum, Akad., VerI, 1009 S.

BERGNER, K; RAUSCH, A; SIHLING, M. (1997): Using UML for Modeling a Distributed Java Application. In: ForSoft project A1 on "Component-Based Software Engineering" of Insitut für Informatik - Technische Universität München

DEMMEL, M.; AUERNHAMMER, H. (2000): Elektronik in der Landwirtschaft. In: KTBL-Schrift 390

DEUTSCHES INSTITUT FÜR NORMUNG (1997): DIN9684: Landwirtschaftliches BUS System.

LGPL LIZENZ: URL: <http://www.gnu.org/copyleftllesser.html>

MOEN, R. (November 17, 1999): Fear of Forking - How the GPL Keeps Linux Unified and Strong. In: Linuxcare, Featured Article;
URL: <http://www.linuxcare.com/viewpoints/article/11-17-99.epl>

MYERS, N. C. (1997): C++ in the real Worid - Advice from the Trenches. In: Dr. Dobb's Journal, Fall 1997 Careers issue;
URL: <http://www.cantrip.org/realworld.html>

STAFF,1. (February 19,2000): "Giving something back" to the Linux and/or Open Source community. Linux.corn;
URL: <http ://linux.com/jobs/newsitem.phtml?sid=74&aid=7302>

STROUSTRUP, B. (1998): Die C++-Programmiersprache. Bonn: Addison-Wesley-Longrnan, 956 S.

VELDHUIZEN, T. L.; JERNIGAN, M. E. (1998): Will C++ be faster than Fortran? Of: Department ofSystems Design Engineering, University ofWaterloo;
URL: <http://www.acl.lanl.gov/iscope97/papers/veldhuizen.ps>

YORK, D. (2000): In the trenches - Twelve Rules For A Better Open Source Project. In: Linux Magazine, February 2000,
URL: <http://www.linux-mag.com/2000-02/trench 01.html>