The LBS Program Library -
The first Open Source library in agriculture
Structure of the lecture:

1) LBS - an open network of services
   1.1) Communication characteristics
   1.2) Demands on implementation

2) Effects of lacking standard implementation on compatibility

3) Open Source LBS program library LBS-Lib
   3.1) Reasons for development within research group IKB-Dürnast
   3.2) Basic decisions, concept and strategy
   3.3) Structure of LBS-Lib
   3.4) Test phases
   3.5) Management as Open Source project

4) Outlook
LBS - An Open Network of Services

Spangler

Auernhammer

Tractor ECU
Fertiliser ECU
Task Controller
GPS Receiver

physical

logical

speed
PTO
3-point linkage

application rate
work width

GPS

CAN - LBS

physical device
service
connection to service
Providing and Using of Services

Spangler

Auernhammer
Device as Autonomous Agent

Spangl

Tractor ECU

Fertiliser ECU

Task Controller

CAN - LBS

phys log

Autonomous Intelligence

use: control strategy, internal states, external states

decide: setpoint acceptance, actor control, publish interpreted sensor values

SPEED information control

APPL. RATE information control

remote service
interaction induced by local intelligence

local service
interaction induced by remote device

Software implemented intelligence

phys log

Spangler

Auernhammer

fertiliser application

fertiliser application

phys log

fertiliser application

phys log

fertiliser application

phys log

fertiliser application
This diagram illustrates the interaction between various ECUs (Electronic Control Units) in a farming context. The ECUs include:

- **Tillage ECU**
- **Tractor ECU**
- **Fertiliser ECU**
- **Planter ECU**

The diagram shows the flow of information and control between these units, with specific expressions indicating the conditions under which certain processes are initiated.

1. **PTO_F = n_{tractor_construction} \times PTO_B**
2. **PTO_F = m_{tillage,fertiliser} \times PTO_B**

The diagram includes logical and physical connections, with arrows indicating the direction of information flow. The following logical expressions are highlighted:

- **m = n \land speed_{tillage} \cap speed_{tillage} \cap speed_{tillage} \neq \{\}**
- **m \neq n \lor speed_{tillage} \cap speed_{tillage} \cap speed_{tillage} = \{\}**

The diagram also indicates that:

- Physical devices provide services
- Logical devices use services
- Physical devices set setpoint, take information
- Logical devices provide information, accept setpoint

The diagram is part of a process data setpoint interest conflict analysis, with contributors Spangler and Auernhammer.
Summarised Demands of LBS on Implementation

LBS

- Network
  - Distributed dependencies
  - Net of states
- Open
  - Co-operative units
  - No master-slave
  - Unknown attributes (producer, model)
  - Unknown services
  - Unknown dependencies
  - Unknown network configuration

Socrates: „I know only, that I know nothing“
LINUX:
• Same API for different computers
• Open Source kernel adapted by widespread community

Solution for Software Developer:
=> less adapting cost
=> lower price
=> better service quality
Problems Caused by Diversity of Embedded Devices

Variety of Application Interfaces (API)
Adaption for every API $\oplus$ limited number of units

base development cost + base support cost + adoption cost + version specific support cost
(number of units / version)

TOO HIGH FOR COMMERCIAL SUCCESS !!

No commercial reference implementation developed
Compatibility Problems Caused by Separate Implementations

Spangler

Auernhammer

DIN 9684 - LBS
70 % clearly defined
30 % ambiguous defined

Company A
- Interpretation A
  - Implementation Variant A
  - insufficient communication

Company B
- Interpretation B
  - Implementation Variant B
  - insufficient communication

Company C
- Interpretation C
  - Implementation Variant C
  - incompatible devices
Service Interaction Enforces New Software Modelling

**Software:**
- switch-case
- loop
- well defined configuration

**Software:**
- abstraction
- flexibility of interactions
- state based
- standard methods for different services

dependency on services, states, interactions of local/remote services, with dynamic constellation of partners in open system

standard control circuit with known parts in closed system
Automatic Gathering of Process Data based on LBS

* simple Implement Indicator (IMI) to represent thumb implement
* IMI with integration of sensor signals
* georeferenced recording (GPS)
* gather sensor signals from tractor
* central storing with Task Controller

But:
Most of the needed devices and development tools doesn‘t exist.

Decision:
* develop software for missing devices
* create program library as base tool

Future:
Publish program library as Open Source to stimulate commercial
development of LBS devices to ease future research.
• implement the demanded functionality

• realise capability and flexibility with
  enough reserve for upcoming challenges

• prefer suitable concept and programming language to restrictions of to
  simple hardware (e.g. disregard 8 bit µProcessor if no C++ available)

• ease adaption to different hardware conditions

• make distributed code maintainance possible
  with strict modularised design
• analyse LBS (DIN9684) standard

• settle unclear elements in co-operation with standardising group (e.g. mean process data value, system announce)

• create worst-case scenarios for extreme conditions

• acquire modularised design of LBS-Lib and test it against scenarios

• select suitable programming language

• implement and test LBS-Lib
standard unique implementation with identical software

==> compatibility
==> combined development
==> parallel test
  increases stability
==> result > \(\text{input}_1 + \text{input}_2 + \ldots + \text{input}_n\)

==> common strategy for adoption to different target systems
==> distributed adoption open to everybody
Modular structure of the LBS program library LBS-Lib

**LBS System:**
- monitors lists of members and services
- represents local members and/or services

**LBS Base:**
- send or receive base data
- hide value overflow

**LBS Process:**
- automatic management of process data
- preprocess received set points

**LBS Terminal:**
- combinalbe layout elements

---

- **System:**
  - uniform access on hardware

- **LBS:**
  - manage functional components
  - access to LBS interactions

- **application program**

---

- **CAN_IO**
- **EEPROM_IO**
- **Aktor_O**
- **Sensor_I**
- **RS232_IO**

---

not to be change implementation of standard conformant communication

hardware access via interfaces

uniform interfaces

different hardware necessitate adaptations
System Management with \textit{LBS\_System}

Monitor lists for members and services deliver detailed informations about the running system. Tasks for local members and services can be implemented automatically.

Example: announce local member

```
// variable (instance) lbs of central LBS object
LBS lbs;
```

```
// variable with GETY\_POS code of local machine
byte b\_my\_gtp;
// variable with name of local machine
byte str\_my\_name[8];
// tell LBS element-object \textit{LBS\_System} to announce
// a local member
lbs.lbs\_system().create\_identity(&b\_my\_gtp, str\_my\_name);
```

```
LBS lbs\_system()
```

```
LBS\_System
create\_identity(byte& rrefb\_gtp, byte pb\_name[8])
```

```
... ...
```

Process Base Data with \textit{LBS\_Base}

Management of LBS base data for active sending or passive receiving systems. Easy update (active) and request (passive) of values on any desired timestamp.

Example: update and request measured value

```
// variable (instance) lbs of central LBS object
LBS lbs;
```

```
// variable for measured value of the radar sensor
long l\_radar\_sensor\_speed;
// update the radar speed of the LBS element-object
// \textit{LBS\_Base} (activ sending ECU)
lbs.lbs\_base().set\_speed\_real(l\_radar\_sensor\_speed);
```

```
// variable for “real speed”
long l\_speed\_real;
// request actual value (receiving ECU)
l\_speed\_real = lbs.lbs\_base().speed\_real();
```

```
LBS lbs\_base()
```

```
LBS\_Base
set\_speed\_real(long rl\_speed)
lspeed\_real()
```

```
... ...
```
system init

announce machine at LBS

wait

announced?

determine work state from base data:
* 3-point linkage position
* front/back PTO
* theoretic/real speed

working?

update:
* whole time
* whole distance
* wheelslip
* working state

LBS End?

update:
* work time
* work distance
* work area

system stop
• Implementation and test on pure PC system with simulated ECU BIOS

• adaption to the BIOSes of the ECU ESX and the prototype of “Implement Indicator” with intensive tests in laboratory with testbed

• massive field test with georeferenced data recording for more than 100h
LBS Testfacility

1. Terminal + task controller
2. Power supply
3. Tractor ECU
4. IMI
5. Navigation
6. CAN monitoring (CANalyser)

- LBS-BUS
- GPS-signal
- Power supply
  - 2-core to 1-core
  - Junction

7. CAN testbed
   a. CAN connector
   b. RS232 connector
   c. PWM output with LED signalling
   d. PWM output
   e. Interconnection field
   f. Configurable digital or analog input
   g. Configurable analog input
   h. Power supply connector
Process data acquisition

Farmstead: power on – farm boundary (x,y)
Field road: not farm area and not field
Field: inside field with virtual headland (time, consumption, ...)
  - total process
  - main field area
  - headland area
  - virtual headland area
Total: Farmstead + field road + field
Cycles for development of Open-Source LBS-Lib

- Development
  - LBS-Library
    - + Documentation
    - + Examples
  - Analysis
    - Test
    - Spangler
- Standard conformance testing (by DLG)
- Test
- Agricultural engineering companies
- Bulletin board for discussion
- Central project management and co-ordination at the Institut für Landtechnik of TUM
- Interested researchers and users

Spangler
Auernhammer
Open Source Program Library LBS-Lib - Possible Future Organisation

Sourcecode with stable and experimental revision
Bulletin Board with Mailing List for discussion and interacting support (no guarantee for response time of software authors)
Concurrent version management of source code with WWW access
HTML Documentation of API and examples (online or download)
Handbook with general and porting information

Possible free of charge services financed by related project or sponsoring
Possible services available for fee

Workshop and training at TUM or in place of interested company
Project development meetings
Response time limits for treating problem reports or questions

Spangler
Auernhammer
• first stable beta version published as Open Source at the end of October 2000

• analyse acceptance and usability within doctoral thesis

• analyse supporting effort and its financing possibilities

• establish company for commercialised services round the LBS-Lib?
  - decision at the end of the doctoral thesis (around September 2001)