SOFTWARE IN TRACTORS: ASPECTS OF DEVELOPMENT, MAINTENANCE AND SUPPORT

Dipl.-Ing. Rainer Hofmann
AGCO GmbH, Marktoberdorf, Germany

Structure of the presentation

1. Introduction – general aspects of software engineering in agricultural machines

As an introduction let me state some key aspects which look simple and are basic to any information technology. But being the driving factors in agricultural machinery technology it may give some transparency to its processes.

In modern agricultural machines with electronic network systems software controls almost all vital functions. In addition to this legacy any technical innovation in these machines is driven or at least accompanied by software. This situation creates an enormous dependency of the manufacturer to all aspects of software creation. Thus being key and core competence for a manufacturer of agricultural machines all intellectual properties should be kept inside the company to stay competitive.

Quality of the machine is defined by the quality of its software; any error can lead to a complete standstill of the machine. Therefore it is reasonable to take any means to make software stable and reliable because errors and malfunctions are not accepted like in a consumer product market.

In any machine handled be people and driving on public roads which applies to most of the agricultural machines software has to be considered as safety critical with different levels of severity.

Due to the rising complexity of software time and costs for software development are difficult to estimate and plans are hard to achieve. Balance of efforts and expenses against quality and function is a management challenge.

The most irritating and misleading assumption taken by non-software-people is the statement: "software can be easily and quickly modified - so let's do it". In fact software has to be considered as a component of the machine like a transmission. This presentation should also give some background to this understanding.

The production of software takes place in the engineering departments and there are no other processes influencing the quality and the costs of the product.

The software engineering process itself is defined in time and timing by its validation process.

Last but not least software is a product which may and has to be sold as such also in an agricultural machine like a tractor.

2. Software applications in tractors – present overview

In a general overview the actual state of software in tractors could be split into 5 main areas of application:

1. Basic (tractor) control functions

2. Human machine interface (HMI)
3. Automatic functions
4. Data management
5. Service and support

2.1. Basic control functions

Basic is meant as generic system functions which drive a definite mechanical part of the tractor:

classic functions like
- engine injection control and peripheral systems: cooling system, grid heater, …;
- transmission systems like CVT or power shift transmissions;
- hydraulic spool valves for power lift and external hydraulic applications with load sensing systems including power beyond
- power train controls like automatic activation of 4-wheel drive or differential lock clutch
- PTO: shifting and engagement control

and x-by-wire systems:
- steering: full electronic steering
- brake systems: park brake, brake assist

and basic electric functions:
- lighting, flasher, washer, …
- HVAC control

The main focus in these basic software applications is to increase efficiency in terms of:
- Optimisation and sophistication of the function itself
- Optimised adaptation of mechanical capacity to the required load and given power (= load sensing). E.g. boost power for PTO applications or limitation of power for transmissions
- Ease operation of the machine in terms of manual power pushing a button instead pulling mechanical levers and prevent mental distraction watching all relevant conditions to engage a function or disengage it in time.
- Increase flexibility in production when a function has to be differentiated.
- Supervision of the mechanical components with sensors to reduce wear and prevent damage e.g. through overload or over-temperature.
- Diagnostic functions do not only detect failure on signals but combine different signals and compare to their relevant logic states of the function model. Thus even corrupted signals may be detected.

  The diagnostic and error detection part of the logic functions and the ability to go to a degraded fail-safe-mode to achieve an utmost availability of the machine takes over 50% of efforts in software design and development.

2.2. Human machine interface (HMI)

The drivers interface to the machine gets a very high point of interest in electronic and software design for a tractor. It is not only the key item for a user to identify and personalize with a machine (and a brand) but it’s the main factor to ease the operation of a very complex machine and combination of machines in very complex operating conditions like road travelling with 60 kph or field work with front and rear implements. For engineers it is always the walk on a knife’s edge between simple operation, easy to learn and offering optimisation for any function in specific processes by complex setup menus.
Graphical displays offer new ways for intuitive and context sensitive operation which is known from the PC application. The art is to find a concept in a moving machine where the driver has to concentrate on other processes.

2.3. Automatic functions

The intention to rationalize human labour to perform the work of an agricultural machine leads to continuous increase and improvements of automation software.

- Engine-transmission managements are

a first step to reduce manual inputs from the driver to optimise the tractor performance. The integration of additional control systems of a tractor for optimisation like PTO or linkage control is in progress.

- Headland management systems are automatic sequence recording and replay systems for tractor operation. Automatic sequences are also available on implements and the industry is working to provide the missing link so that a combination of tractor and implement automatic functions would be available (-> ISO 11783 part 14).

- Automatic steering in the field with GPS satellite navigation as parallel or contour tracking is the latest step towards driverless tractors. One of the next steps is to integrate field boundaries and other field or crop characteristics to provide an optimised track pattern which the tractor should follow.

- Integration of tractor-implement process in closed loop control circuits:
  This area is linked to the development of a communication standard ISO 11783 on CAN to exchange data between tractor and implements. Current industry developments focus on integration of display and operation panels of the implement (refer to ISO standard 11783 part 6 virtual terminal) in standard terminals in the tractor and the use of basic messages from the tractor.
  Next steps in realisation: sensors of the implement process control tractor functions e.g. steering, speed or linkage position.

- Integration of field and crop process data to control field operation = precision farming. Neither online control systems with nutrient detectors (real-time approach) nor the mapping approach have yet achieved a penetration of the market.

2.4. Data management

Latest developments in society speed up this area of software in agricultural machines. The requirement of traceability of food and its production process, documentation of environment sensitive works like spraying pesticides or applying fertilizer. In addition farm work has diverted, the driver of the machine combine or tractor, is no longer the owner of the process. He is either an employee as tractor driver or a contractor and not involved in the farm process of crop production with all its relevant parameters and control processes.
This area covers a wide spectra of software split into the parts

- machine related data acquisition and logging
- office software for documentation and further processing of data.

Machine related data processing can be separated into 4 areas:

  - Information about condition, performance and load of a machine: e.g. working hours, speed, fuel consumption, oil condition, …etc.
  - Data from the agricultural working process: e.g. acres of covered area, yield, crop density, soil compaction, … etc.
  - Driver or labour data: e.g. start of work, break, kind of work, …etc.
  - Process data of used substances and the flow of material: e.g. applied mixture of chemicals in fertilizer, … etc.

At this point exact and fixed interfaces between data process on the machine and the office software is basis for cooperation between machine manufacturers and office software suppliers. This pays back regarding the fact that almost each country in Europe has its unique documentation system.

2.5. Service, support and manufacturing

A wide range of software activities are necessary to provide service and support for complex machines like modern tractors.

2.5.1 Service

Onboard diagnosis has been mentioned as part of tractor control software whereas off-board diagnosis PC systems read out error codes and visualize all states of electronic signals, logic states and communication messages or log trace files. The communication to the tractor electronics via CAN bus could be based on standards like SAE J1939 or KWP 2000 (key word protocol) an automotive derivate.

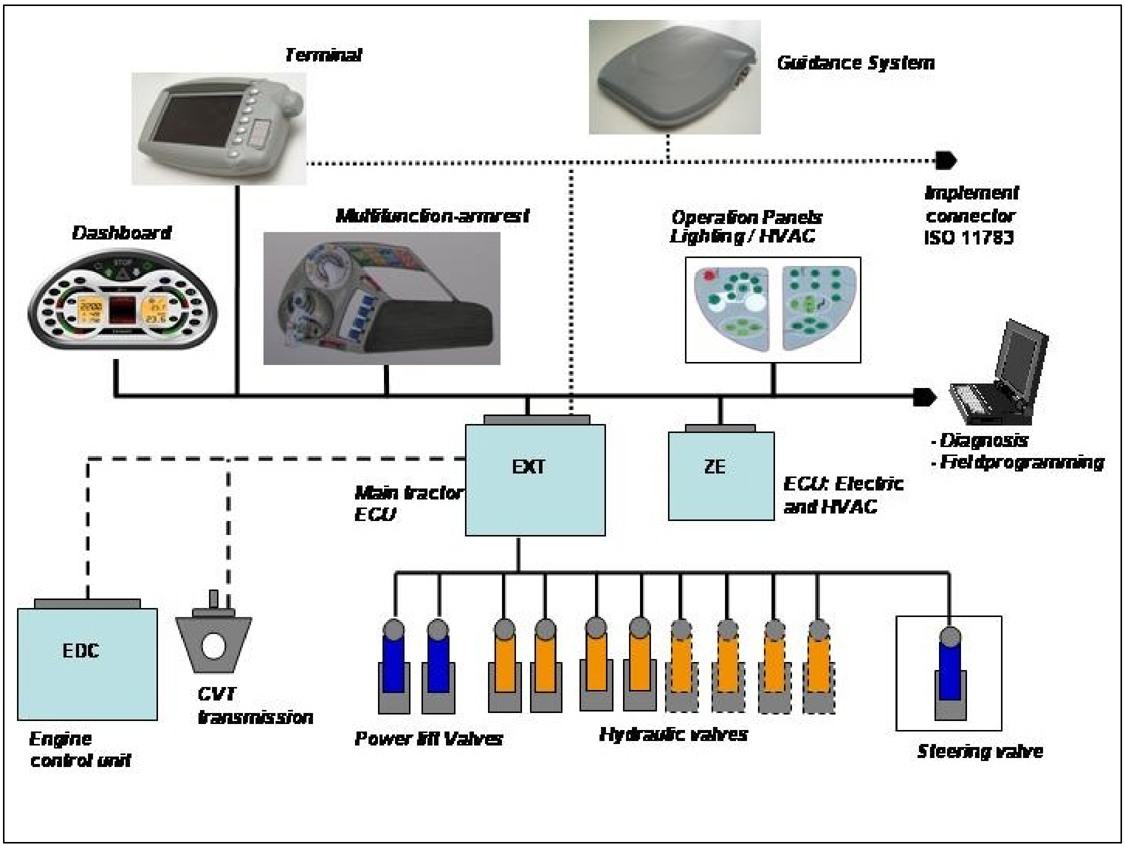
2.5.2. Manufacturing

Flash technology in hardware memory has created a new phase of flexibility in machine production. Software is handled as an own individual part of the tractor (machine) sent to production at assembly time like a bolt or a wheel. In addition huge sets of parameters for each function allow configuring the tractor individually.

2.5.3. Support

Fieldprogramming software allows easy treatment of spare parts as ECUs can be stored without software and programmed with the latest software and sets of parameters specific to the type, model and unique equipment of the customer tractor. The complex background of this system will be explained in a following item.

2.6. Examples for software control networks in a tractor



Complexity is sometimes easier visible in figures and dimensions.
To illustrate the state of software complexity in a standard tractor (top class model) a few figures:

- Network consists of 4 CAN-Busses
- 20 ECUs or subsystems with microprocessors
- Software per ECU:

Multifunction armrest: 16.000 lines of code (50 files)
Dashboard: 28.000 lines of code (94 files)
Main tractor-ECU: 120.000 lines of code (230 files)
Graphic Terminal: 160.000 lines of code
Graphic Terminal with Linux operating system: 50.000.000 lines of code (OS) and 250.000 lines of code (application

- More than 90% of this software is written in C-code.
- Software development is shared between OEM-Manufacturer and suppliers e.g. hydraulic components approximately 50% – 50%

3. Economical requirements for software design in agricultural machines

3.1. Challenges for software design

Software design and development process in agricultural machines is challenged by different facts:

- Agricultural machines generally have low volumes compared to automotive or even consumer electronic products.
- Product differentiation is not only a marketing aspect. It is required to optimise the machine function to specific international markets with different farming methods and down to adapt to unique conditions of soil or crop.
- Electronic hardware components like μ-processor or memory provide short life cycles compared to the very long product life cycles of a tractor being in production for 10 … 20 years and a given spare part option of 15… 20 years after end of production.
- A tractor is a safety critical system in general. Any software controlled function has to consider hazards of user and environment during machine operation and in public road traffic.
- As software functions may be manipulated it is an obligation for the manufacturer to provide successful means to protect critical functions e.g. max. speed control or engine power
- Costs of electronic hardware decrease compared to its performance.
- Hardware costs add to the price of each tractor, software are one time costs.
- Reusability of software is an economic must.

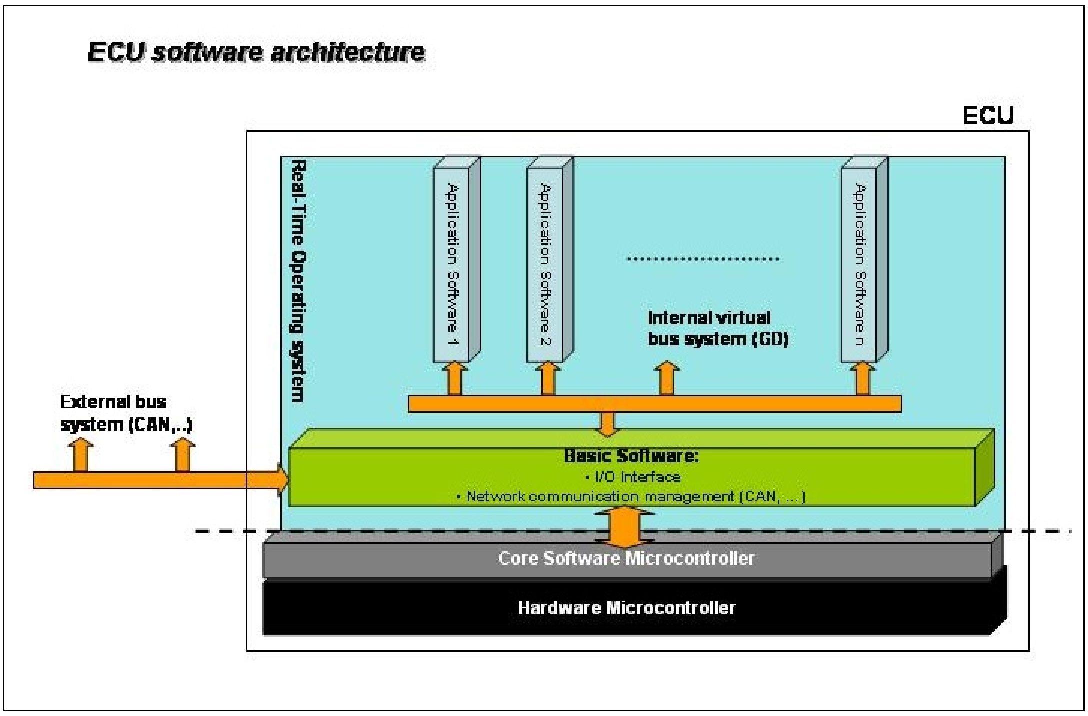
3.2. Strategies and principles for software design

These challenges have lead to following strategies and principles in software (and hardware) design:

ECUs are designed to provide hardware independent software with a high density of application modules.
A base block called core software handles all microprocessor relevant and specific parts of software.
A second basic set of common service software handles all in- and output signals and communication buses and is designed more unique to the ECU type of signal and bus hardware.
A real-time operating system provides a time scheduled structure for all software modules using the single processing channel of the microprocessor.
The key for reusability of software modules is a generic communication system with a standard set of messages which is established internally between the modules. It is sometimes referred to as virtual bus system.
The communication between ECUs in the tractor is based on a network topology of CAN bus with a generic or standard communication protocol. A general industry standard is given by J1939 or ISO 11783 but also proprietary solutions are available.

If internal and external communication structures are identical application software modules are independent of their physical location and may be shifted from one ECU to another with minor software adaptations. Application software modules may be handled as encapsulated parts and offer a high flexibility for the hardware and network layout of a machine. This enables also the integration of supplier software in different ECUs in an economic feasible way.

A typical structure of ECU software architecture:

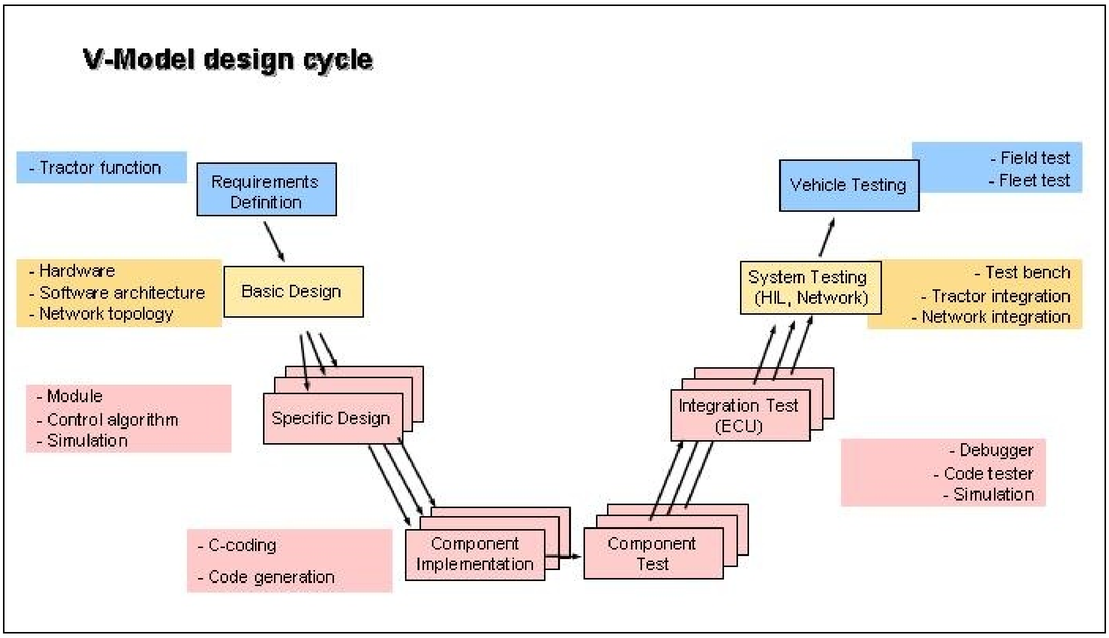


In parallel to these architectural structures a defined and detailed software design and verification process is prerequisite for a successful integration.

4. Software development process

The V-model is an industrial wide acknowledged process guideline for software design and verification. It describes top down design steps and its equivalent bottom up validation steps with the level of detail growing to the bottom point of the V.

This model provides guidance for the management by demanding an individual handling of each step closing the previous one before entering into the next level. In addition each closed step offers the opportunity to specify all relevant test procedures for the validation process on the equivalent level in parallel.



4.1. Software design phase:

Typical steps are:

- Functional specification
- FMEA
- Software specification
- Network and communication layout
- Simulation, rapid prototyping
- Manual code generation or C-code generator

If safety critical functions have to be considered a risk analysis has to be prescheduled before starting the concept phase. Results of this analysis may lead to a differentiated layout of soft- and hardware structures e.g. parallel path or multi-path signalling and processing. Software design relevant to safety critical functions gets a growing focus under the light of product liability and is handled in emerging standards like ISO 61508 which are currently condensed to practicable use.

Growing importance achieve software tools CASE (computer aided software design) which channel this process and provide consistency from concept to C-code including documentation. Most importance of these tools is focused on simulation of control algorithms and its rapid prototyping by C-code generators. It should not be overestimated as still more than half of the code consists of error detection and fail safe mechanisms which are difficult to simulate.

4.2. Software validation phase:

Validation is the most critical path in software development:

- ‘You never know when it is really finished’.
- ‘You never get software tested to the status of error free’

Experience shows most efficiency in multi-path testing i.e. diversifying test methods as well as persons executing the tests and their different view onto the software.

Automation of test systems for software is beneficial if test scenarios are 100% repeatable. This is valid for hardware testing, basic software (I/O, communication) or to replicate module tests after modifications. For application software the effort to write test scenarios is rather high and vehicle integration tests seem not to be replaceable in a viable way.
Automation takes place in terms of simulation of communication networks for components to replace static tests on tractors.

Typical steps in software validation are:

on module level

- Manual code testing with debugger and emulator
- Automatic code checking tools
- Hardware-in-the loop(HIL) test benches for simulation and stimulation

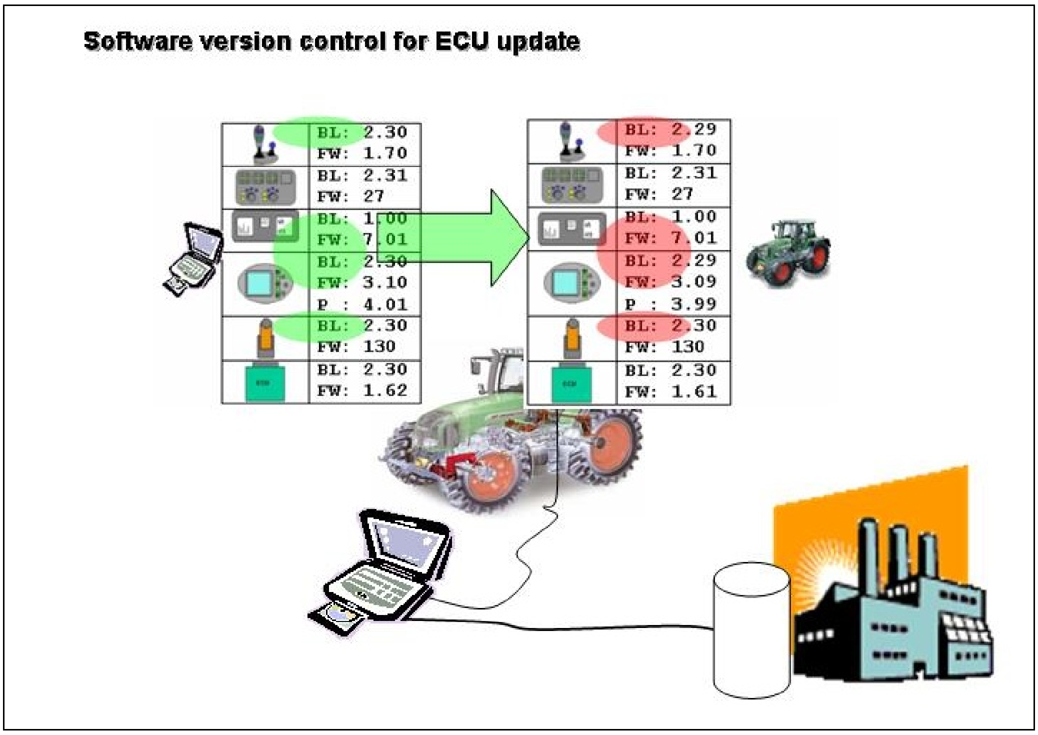
on integration level in tractor:

- Validation of functionality
- Validation of FMEA i.e. simulation of signal failures and error states and proof the specified reactions of error detection and fail-safe state
- Field test
- Fleet test

In the end often a crucial question has to be answered: ‘Does a modification of one line of C-code require the full test cycle again?’

5. Software maintenance and update

Software changes are part of lifecycles of a machine. Modifications are required for optimisation of a function, extensions and mechanical modifications in a system, redesign of hardware or just fixing software or errors which have not been detected during the test phase.
Any modification of software has to undergo the same validation process like a new module. The crucial part: this modification of one (or more) module(s) has to be integrated into an existing network of software modules.
The quality rule is strict: ‘Any update of one module has to be tested for consistency to any other module in a tractor.’
Considering the numbers of ECUs in a tractor which also change their hardware during lifetime and the density of software modules and considering that most software modules are regularly changed, a test of any module against all other modules and their versions and their updates is not achievable.
Therefore only defined combinations of module versions (and ECU hardware versions) are checked as an integration test in the tractor. These released packages of software versions are allowed for updates in a customer tractor.
Software version control:
To achieve a qualified update process in any service station a database software system is necessary to manage the version control of all relevant software modules in all tractor types and models.
Basis is a definite identification of each software module, each parameter data set and the relevant hardware (ECU).
Labels are defined as released combination of software versions over the whole network of a tractor.
In a service case the system may read out the given set of software versions in a tractor and compare with those in the data base. On request it uploads all relevant new modules which belong to this released package (label). An individual change of software without that system is no longer possible.



6. Software as a product

Considering the mentioned new technologies like flash programming and software methods like generic communication and generic architecture of modules as well as database systems for updating software in the field, software in agricultural machines becomes a new touch: it is marketable as an option and as an add-on for a machine like a tractor. It is difficult to understand for a customer in a traditionally conservative market that he should pay additional money where no real part is visible. Nevertheless habits from consumer market products are adoptable and it should be explainable that a headland management system is only a piece of software.

This creates new complexities and challenges for organisation and software development:

- Identification and registration of individual software equipment of a customer tractor
- Commercial release process for a software module update on dealer level
- Protection against copy
- Logistic processes e.g. to provide software via internet

7. Manufacturer integration process

The development of a common standard for data communication between electronic control systems of tractor, implement and office software is one of the big milestones in the history of agricultural engineering. Starting with LBS ('Landwirschaftliches Bus System') and DIN 9684 it has been reworked towards the international Standard ISO 11783 and labelled as 'ISOBUS'. In spite of its hesitating introduction into products on the market it shows its real benefit on long terms.

Main focus of this standard has been:

- open system communication structure for CAN network
- graphical display as virtual terminal
- data dictionary as interface to office software

To be developed:

- service and support: standards for diagnosis tools and software version control
- further integration of automation between tractor and implements
- models and 'best practise' for manufacturer overlapping software engineering processes comprising validation and release and

synchronizing updates in the field

This is evident considering the manufacturer internal validation processes which have to be applied for components in a network of a closed system like a tractor or combine and projecting it on an open system where different machines of different manufacturers have to be integrated in a common network.
First business years of a neutral test institute (DLG in Germany) in testing and certifying compatibility of electronic components to the standard shows an economic discontinuity.

This approach would require a platform organisation for manufacturers, suppliers and test institutes to coordinate and synchronize activities for software validation and release and provide neutral test arrangements for machine integration tests.
German manufacturers association for agricultural machines has started a process to establish such a platform worldwide.

The automotive industry has established a similar standardization process in an manufacturer driven organisation called AUTOSAR (**AUT**omotive **O**pen **S**ystem **AR**chitecture). Its task is the definition of structures and interfaces for software modules to provide the option of using similar or identical software modules in ECUs. (reference: www.autosar.org)

8. Conclusions

It is not a prophetic act to predict that software and its peripheral aspects will tremendously grow in agricultural machine industry.

Functionality of software
The need to produce agricultural products with economic methods will keep up the demand to increase efficiency of the machines and reduce input of man labour.
On one hand the user (driver) of a machine should become able to run as many functions on one machine as possible. On the other hand he will be enabled to run and control additional unmanned machines in parallel (remote control).

The control software of a machine will be embedded in a process-software controlling the whole agricultural procedure like soil preparation, planting, fertilizing or harvesting.

Field robotic with totally unmanned machines will become likely in closed areas.

Complexity of software
It is obvious that the complexity i.e. the number of modules and their interdependencies will increase dramatically and continuously.
This complexity will overlap tractors, implements, combines, etc and office software and following process software for food production.

Engineering efforts and expenses will grow in the same extend:

- … for design and development of software
- … for test and validation of software
- … for administration and support of software in production und service

What should or could industry do?

'Think in processes' i.e. to adapt software design and development methods for agricultural machines to those used for big software systems like in automation industry or data base systems. How do other industries like telecommunication or finance systems handle their enormous software systems?
Improve methods for absolute encapsulation of software modules so that modification of one module does not influence others.
Develop methods in software to make it robust to its own failures: like in biologic systems, health is achieved by increase of

immunity.

Manufacturers can not afford to develop all software on its own and in competition to all other manufacturers.
Standards like 'ISOBUS' (ISO 11783) should be extended and basic software libraries could evolve being used by all manufacturers.
This requires also economic and management models and platforms for software development processes between manufacturers and their suppliers.

In the end an old (in terms of software development history) idea appears:
'Let's create an open source community'?
Maybe not but at least an open architecture is necessary.

References

[1] Auernhammer H., *The role of mechatronics in product traceability*. 2003, Club of Bologna, topic 1.2.a

[2] Waldmann, A. *Kontrollierte Software-Updates von elektronischen Fahrzeug-Steuer-geräten*. 62. VDI-MEG-Tagung ‚Landtechnik', 07./08.Oktober 2004 Dresden,

[3] www.autosar.org; AUTOSAR_LightVersion_V1.5_f.doc: *Automotive open system architecture – an industry-wide initiative to manage the complexity of emerging automotive e/e-architectures.*

[4] Heinecke H. et al. *AUTOSAR – current results and preparation for exploitation.* Euroforum conference, May 3rd 2006, Stuttgart

[5] Hieronymus P, Hofmann R, *Kommunikationssysteme*. Jahrbuch Agrartechnik 2006, Kapitel 2.3

[6] ISO 11783: Tractors and machinery for agriculture and forestry – Serial control and communication network, part 1-12