Abstract

Design of complex systems involving a number of experts from different fields always includes modeling at many levels of abstraction. Still, each modeling methodology, including model-driven approach using UML, requires partners coming from an area different software engineering to learn a new language and adapt themselves to the new methodology. As this can be tedious and even contra effective with respect to efficient project management, we propose an adaption and a simplification of this, traditional, approach. We believe that the approach is easy to be understood and applied but still results in an useful model that can serve as a mutual understanding platform. We present the application of the methodology as we achieve a full and detailed structural model of infrastructure for precision farming. Modeled infrastructure is an open ICT hardware-software solution based on ISOBUS specification that enables partial automation of tractors increasing safety and production efficiency.

1. Introduction

The constant population growth in developing countries followed by the increased food needs are pushing toward more efficient, safer, environmentally friendly and sustainable food production [9] [13]. Precision farming (precision agriculture), relies on usage of Information and Communication Technology (ICT) to identify, access, monitor, manage and evaluate the spacial and temporal variability of agronomic parameters. Mostly related with site-specific fertilization its influence spreads in all areas of agriculture [4]. Major drivers of precision farming are positioning system (GPS), yield mapping and distributed sensing and control systems [9, 5]. In order to achieve high production efficiency interoperability and safe communication between those and other ICT components deployed on tractor and implements are necessary. For that reason, based on Landwirtschaftliches BUS-System, ISO11783 (ISOBUS) communication standard providing communication and control network and defining basic ICT components has been defined [12]. This way, instead of having propriety interface between tractor and each separate implement that are connected with a number of different cables, the same control and user interface can be shared between standardized implements and tractor.

This paper presents a model of an infrastructure for precision farming based on extension of ISOBUS that provides more functionalities and increases safety. Due to the complexity of the system and involvement of partners coming from different areas of expertise (both academia and industry) we chose to adapt and apply the model-driven approach as it has been already successfully applied in similar projects [17, 8]. Still, we find that the approach is not necessarily well accepted by engineers who are unfamiliar with software development, and in some cases it can cause irrelevant discussions and confusion. In addition it requires from all the involved parties to learn appropriate modeling language (usually, Unified Modeling Language (UML) [11]) and to fully adapt their present approach to the new methodology. For that reason, we have simplified the approach and adapted it to the particular case presented in section 4.

The material reported in this paper has been developed in the framework of European Project STRATOS [2], in the ERA-NET ICT-AGRI [1]. As a matter of fact, the expressive characteristic of UML has been fully exploited in order to allow a tight cooperation between the 6 partners, from 5 different countries, and with different academic and industry backgrounds. The results achieved and here described testify a success story about the use of UML for system development in agriculture.

The rest of the paper is organized as follows: Section 2 gives an overview of the ISOBUS standard as it is the core of the infrastructure proposed. Model-driven methodology and proposed simplifications are presented in section 3, while section 4 presents the model design process and the final structural model of the system. Finally, section 5 concludes the paper and gives the outline for future work.

2. ISOBUS standard

A tractor or an implement taken apart are useless, as only the combination of the two of them performs an effective agricultural job. In the past, all tractors had a pro-
proprietary hardware architecture to control implements, and in particular an ad hoc remote control system which must be installed in the tractor cockpit, often installed as an aftermarket kit. Since many implements should be plugged into a single tractor to accomplish various farming jobs, many control and user interface (HMI) devices have to be installed on board often producing a mess of cable and hardware, that prevent ergonomic and safe tractor use. Since the early 90’s a task force of tractor and implement manufacturers, and standardization authorities is developing the ISO11783 norm (ISOBUS) [7], that aims at introducing a “plug & play” concept into the tractor-implement systems. We give a set of components relevant from these papers point of view that ISOBUS specifies for tractors provided with ISOBUS certification [14], [12]:

- Virtual Terminal, a standardized control and user interface unit that enables communication of control and supervision signals with any ISOBUS-compliant implement. In such a way, all standard implements can share the same control and user interface unit, simplifying by far the work of the farmer and increasing the usability and, hence, the safety of the tractor.
- Task controller, a software application run by Virtual Terminal for the management of the agricultural job.
- Communication network, based on 2560kbaus CAN communication protocol allowing simultaneous bus access from different nodes.

3. Adapted Model-driven approach

Model-driven approach is already being extensively used in design of modern software but also complex systems that can include hardware, software, physical components, personnel, etc [16] [17]. It is practically the most powerful and the only mean for handling system’s complexity. Typically, the model-driven design is an iterative process having following stages [17], [8]:

- Context description, exposing a very generalized system structure and identifying the surrounding elements and systems, actors and stakeholders.
- Requirements engineering that includes: collecting, tracing, analyzing, qualifying and managing user requirements (functional and non-functional ones) as well as defining system requirements based on those [6]
- Use cases determination that includes definition of system functionalities and user scenarios based on previously analyzed requirements. Use cases are usually organized in hierarchical manner meaning that high level use cases are defined first while fine grained ones are defined in later stages and usually directly mapped with system requirements.
- System modeling, that includes development of structural and behavioral model of the system according to the use cases. This stage itself has to be done in several iterations. System modeling usually starts at a very high level with the generation of a general structural model based on context description, and mapping major use cases to system components while describing behavior of the system for each use case. Once all use cases are mapped, described and the structure adapted accordingly lower level use cases and internal structure of components are modeled in a similar manner. Finally, system is being validated and verified.

Despite the success of a model-driven approach and its wide application, based on experience from several international joint projects that have included experts coming from both academia and industry, we find that the direct application of a model-driven approach could be difficult. This is particularly true when it comes to partners who are not familiar with software development and principles of object-oriented programing. On top of that, in many cases, the desired system should not be developed from the scratch but it should include existing components, algorithms and methodologies already being used in other, perhaps, similar systems. In that case, models already present in other languages should be re-described in details in UML. Still, we find that simplified model-driven approach and especially developed structural model can be efficiently used as a mutual understanding platform. Such approach should provide a general system model keeping in mind particularities of each part developed by a different partner while internal model of components should not be mandatory developed in the same manner. Actually, a partner involved in the design of a particular component should decide on design approach while keeping with agreed interface. In addition, we find that, in cases when project goals are already defined but (like in case of research projects) partners still have some space for adapting system description in later stages; requirements engineering part can be omitted and use cases directly defined from the system context.

Having in mind all previously mentioned, we proposed a simplified model-driven approach as depicted in figure 1. Major differences when compared with previously presented general stages of model-driven approaches can be summarized as follows: New definition of context has been introduced; requirements engineering has been omitted as use cases can be directly extracted from the project description; structural modeling is explicitly done before behavioral modeling so the system structure can be achieved as soon as possible. In addition, we assume that each partner is already well experienced in the design and development of parts he is responsible for, and thus behavioral model can be even omitted as well. If that is the case, we believe that this, much simplified approach can provide fast, but still good results when compared with the typical approach and straight application of all
the stages. In particular, we believe there is no need to “reinvent” already existing components and to insist on all partners to apply the same approach, but the result of modeling should be an overall model that will serve as a mutual understanding platform.

As a modeling language we chose UML [11] profile for systems engineering, SysML [10], as it has been already used for modeling systems similar to the one that we have in mind [15]. SysML extends a subset of UML and provides additional diagrams that allow modeling of software, but also hardware, personnel, different kinds of flows, etc. Thus it is more appropriate for modeling ICT systems in agriculture.

Figure 1. Simplified and adapted model-driven methodology

Methodology presented in figure 1 starts with the context description that usually can be directly drafted from the project description. We believe that the usage of UML/SysML is not mandatory here since the resulting model of this step should just give a taste of the future system structure.

Use cases definition should be done in a hierarchical manner while major use cases are extracted directly from the project and context description. Once an agreement has been achieved about major use cases, fine grained use cases should be proposed by partners responsible for the development of particular system functionalities. It is very important to take into account also non-functional requirements that can be presented as constrains related to use cases. This step also includes identification of external actors and their relation with use cases, while major components could be annotated in use case diagrams as well.

In the next stage, the initial system structure given in the context description should be refined based on the use cases defined (describing system services and functionalities) and each use case should be mapped to a particular set of components. This is an iterative process that could even require revision of the previous steps. Once major use cases have been mapped to components, the internal structure of each component should be defined and fine grained use cases mapped. Still, this second part should be up to the expert/partner responsible for component development as he could choose to use other mean for modeling and designing of a particular component. The major result of this part is a clear idea on partners responsibilities, components relations and interfaces, having all the system functionalities and constrains clearly defined. We find this to be the most important result of applying the methodology as the mutual understanding platform can be achieved without the necessity of understanding each step of a traditional model-driven approach and with minimal knowledge of UML/SysML.

In next step, modeled components should be integrated and behavior of the system defined in a more precise way. Even that components were modeled according to defined use cases and constrains, this step could still introduce potential problems. It is likely that minor revision of the system structure, and modeling of components themselves will be required after a more precise modeling of the system behavior.

In the final step, validation and verification, we check if the system is operational and in-line with the initial requirements (use cases and constrains). If we find this not be true, the system model needs to be reconsidered from the very beginning.

4. Case study

In this section we will show how the previously described methodology can be applied to the real case of an agricultural project. We will also present a structural model of the envisioned system. Our goal is not to present all the methodology steps in detail but rather to give an overview of the methodology application, to prove how it can, in a simple way, lead to the definition of the final structural model. Parts of the methodology, that are find to be more specific to the particular project, like components integration and system verification will not be presented in this case study. We will first, in brief, expose the system goals, then we present in detail use cases and show how we relate them to system components. Finally we propose a structural model of the system.

4.1. System goals and context

Our major goal is to model an open ICT hardware-software infrastructure for precision farming based on the ISOBUS specification [12]. The infrastructure should enable the partial automation of tractors and at the same time enhance their operational safety and production efficiency, with the positive effects of reducing of accident risk and environmental impact [2].

In particular, the system will be based on the ISOBUS
(see Section 2) compliant, wireless self-powered sensor network (having acceleration and temperature sensors) for the real time measurement of soil and harvester conditions. In this way, Task Controller can optimize the whole tractor and implement operational modes to improve the farming job quality and safety of the overall systems.

Particular attention will be given to the system safety defined through the set of constrains that will be presented later (in the scope of use case diagrams) and advanced algorithms to be run by the task controller.

Having this in mind, system context can be modeled as depicted in figure 2. We present here, on purpose, the initial version of the context description that is not necessary, as we will see in next subsections, fully in-line with the final model. In our opinion, context description should only provide a general idea and serve as a starting point for further modeling. The system consists of a stationary part (Farm Infrastructure) and a non-stationary part implemented on tractor and implement(s), that includes controlling logic, communication network, WSN and positioning system. Details of system context will not be discussed here since system components and architecture will be described in the next sections.

4.2. Use cases engineering

As already stressed (see Section 3) we make use case engineering in a hierarchical manner starting from the high level as depicted in figure 3 with the description provided below. Constrains, as non functional requirements are described directly in the figure.

**Figure 3. High level use cases set**

**Precision farming** is the most general use cases giving major project goals as already described in the previous subsection.

**Set up working parameters** refers to setting up parameters (in real-time) for the tractor and the implement based on the application of the algorithm applied to the real-time measurements and historical data. **Calculate new parameters** includes the calculation of the parameters to be set up. **Collect data** includes gathering of real-time measures through envisioned sensorial system from the environment. The system should be able to store and, when required, acquire the data. Real-time and historical data will be used with the algorithm for precision farming defining tractor and implement parameters. **Communication** refers to all communications that cannot be done though ISOBUS network. In particular, communication between
the on the field system and the Farm information infrastructure that stores historical data can be realized using GMS network as external component modeled as actor.

As a demonstration, in figure 4 we present the decomposition of Collect data use case giving more details about related external actors, system components and constraints.

In a similar manner, all use cases were decomposed, described in details and related to system components. By the end of this process, we get a mature set of system components to model system structure. This will be presented in next section.

4.3. Structural model

The structure of the system that we envision is relatively complex, therefore we will present the model in a hierarchical manner. That way, each partner can have a general structure overview while focusing only on the part of the structure that he is interested in. We present the model of the structure, on each level, using SysML Block Definition Diagrams (BDD) and Internal Block Diagrams (IBD) [10]. While BDDs present the system model in a more abstract way, giving components (blocks), their relations and dependencies without showing exact connections and ports; IBDs give more a detailed view on the structure showing instances of blocks (parts), their ports, connections, etc.

High level system model has been presented in figures 5 and 6. STRATOS context is an abstract block that serves as container for all other system blocks and does not present component to be instantiated. ISOBUS components, described in section 2, are embedded inside different system blocks. Block are defined based on their functionalities and deployment location (tractor, implement, etc.)
work provides a mutual communication mean between system blocks and internal components.

Figure 8. Detailed model of on board infrastructure

The model of the on-board infrastructure is depicted in figures 7 and 8. It consists of: the ISOBUS Task Controller that supervises sensor data acquisition, controls the farming job, communicates to Farm infrastructure the data acquired and receives the job parameters form the Farm infrastructure; User interface that uses the interaction primitives (graphical symbols, input devices, etc.) of the ISOBUS Virtual Terminal (VT) to permit the human supervision of the STRATOS on-board infrastructure; Sensor access point that collects data from wireless nodes.

Sensor infrastructure unit model is presented in figures 9 and 10. According to the system description, it should be capable to harvest energy and collect information about temperature and acceleration. Modeling of power supply of the sensor is done according to [3]. Harvesting units implementation varies depending on the place where the node operates: if it is on board of the tractor or an implement, the vibrations of the vehicle can be exploited to power the node, otherwise the energy provided by solar radiance and temperature gradients can be used. Communication unit provides connection (directly or in a multi-hop fashion) with a wireless gateway, using Bluetooth or a protocol designed for low power and short range devices like ZigBee. Processing unit operates the node, prepares the data collected from sensors to be send to the Task controller and keeps the configuration settings of the node.

Figure 9. Abstract model of sensorial system infrastructure

5. Conclusion and future work

In this paper we have presented how the model-driven approach can be successfully adapted (simplified) and applied to the areas that are not purely software design and ICT structures but involve other elements as well. We have applied the methodology in the field of agriculture, for modeling the ICT infrastructure enabling partial automation of the tractor for precision farming. As designers and users of such infrastructure are mostly coming from the area of systems automation or agriculture, we found that the direct application of traditional model-driven approach might be unfeasible. The main reason is the unfamiliarity of designers and users with details of the approach and used modeling language, UML. For that reason, we have proposed and applied a very simplified ap-
approach clamming that its application can provide a model of a system structure in fast and easy manner. We prove this claim while presenting, in details an application of the methodology to the envisioned infrastructure starting with the project description. This model has been developed in the framework of the European Project STRATOS, that includes 6 partners, from 5 different countries, and with different academic and industry expertise. The results achieved prove a success story about the application of the proposed methodology.

6. Acknowledgments

This is to thank to the STRATOS project partners for carrying on work presented here and to the EU ERA-NET ICT-AGRI program for their support. Finally, we would like to thank the national founding and supporting institutions (alphabetic order): Latvian Academy of Sciences (Latvia), Ministry of Agricultural, Food and Forestry Policies (Italy), Ministry of Agriculture and Rural Development (Israel), and Swiss Federal Office for Agriculture (Switzerland).

References