MDA Compliant Concept for Maintaining Embedded Multifunction Displays

Q. Zhang and A. Karcher, Member, IACSIT

Abstract—This article provides a holistic concept for a Model-Driven Architecture (MDA) compliant embedded multifunction display (MFD) development addressing the challenges of MFD SW maintenance. Embedded MFDs show information from their parent system in form of symbols, providing clarity and user support. Especially in persistent systems such as a helicopter MFDs demand constant maintenance and adaptation. Because of the concentrated display of system activity the MFD software (SW) has to face MFD-specific and display-relevant system changes. Since the MFD is highly dependent on other systems, display relevant system changes cause a MFD SW modification. Therefore MFD SW maintenance has to take the embedded character into account, e.g. how changes in information of display relevant systems are adapted to the MFD symbols. Current development models for embedded systems encourage partitioning the systems into further (sub-)systems and designing them independently from each other for parallel and time-saving development. In case of MFD symbol modification caused by other systems, however, dependency is desired, and current approaches based on independent subsystems give no clear instructions how and in which development stage the MFD will get the information, making the SW maintenance process time consuming.

Index Terms—Multifunction displays, embedded systems, SW maintenance, display symbols, model-driven SW development.

I. INTRODUCTION

The integration of humans into a mechatronic system, e.g. helicopter or car, becomes more and more important in modern systems. The display of system information and the event handling of user actions are increasingly implemented by configurable [1] MFDs. In the beginning MFDs were used in military helicopters since they concentrate information of the flight system e.g. weather, navigation or fuel consumption, in one device. By replacing of corresponding display elements of the respective instruments place was saved in the cockpit [1]. In persistent systems, MFDs may operate for decades, during which they face several changes. The changes can either affect only the MFD (e.g. the customer wants a symbol modified) or system changes causing the modification of the MFD SW (e.g. adding a new system component whose information has to be displayed at the MFD). Because of its special characteristic of being dependent from other system states, the MFD SW cannot be processed on its own without consideration of other systems. In order to show why MFD SW maintenance is still challenging the following use case of MFD SW modification caused by a (sub-)system change is presented based on real occurrence.

For the helicopter to operate in a different environment, a dust filter is required in the cockpit whose states are to be presented to the pilot by the MFD. How should the new filter symbol be parameterized – is there already an existing filter symbol or is a new one to be created? Where to integrate the new symbol? Those MFD graphical decisions depend on the specification of the filter system (for symbol parameterization) and the location of the new filter system within the cockpit (important for a reasonable integration of the new filter symbol in the existing MFD formats). That means the sooner the filter requirements are given to the MFD the faster the MFD modification can begin. From the process model view both MFD and dust filter are (sub-) systems of the helicopter. The dust filter is developed from scratch and all its system, component and design specifications are defined. Some of them are MFD display related, e.g. state (system level) and parameterization (component level). The MFD already exists and only the SW component responsible for the symbol generation is affected by the modification. How does this display related filter information get to the MFD and can both the MFD and the dust filter modification procedure be executed in parallel for more efficiency? Current embedded system approaches do prefer the early separation of subsystems for parallel development. Other GUI methods focus more on the model based development but neither of them provides a satisfactory solution for maintaining embedded MFDs with a holistic view of the superior system. This article represents a holistic concept for improving embedded MFD SW maintenance with focus on the symbol modification.

II. THEORETICAL FOUNDATIONS

Despite intensive research, there are no approaches found providing a satisfying solution for all the problems mentioned in Chapter I. Either the approaches focus on model based SW development of GUIs, or current works take development processes and design of embedded systems into account without the emphasis on the special characteristics of MFDs. Furthermore, both research directions concentrate their view on the development and not on the maintenance phase.
The Cameleon Reference Framework [2], [3] provides a concept of developing user interfaces (UIs) according to Model Driven Architecture (MDA) approach [4]. It proposes four different levels of UI abstraction (Task & Concepts, Abstract UI, Concrete UI and Final UI) being modeled by UsiXML [5]-[8]. The idea is to first define tasks of the UI indicating logical activities on the “Task & Concepts” level, and then group those tasks and defining interaction spaces in the “Abstract UI” phase. The look and feel of the elements in those interaction spaces are set in the “Concrete UI” level where code is then generated for the “Final UI” [5]. The representation of those symbols (Concrete UI) can vary from platform to platform. For example, a button symbol looks different when transferred to a Java application or a MacOS –X platform [6].

Fig. 1. UsiXML models according to CRF [7].

The levels of CRF are too vague when it comes to incorporating the platform model according to the MDA [8]. One possibility is shown in Fig. 1, where the levels of CRF are extended by a new level “context of use”. Here, all platform information is defined by the contextModel needed for the development of the transformationModel and mappingModel.

However the CRF only focus on the optimization of MFD SW development, but its concepts do not provide sufficiently detailed information to derive graphical MFD specifications that minimize current challenges of MFD symbols modification. As mentioned in chapter 1 it is not enough to concentrate on the MFD itself since the MFD is embedded and SW maintenance is also caused by system changes. According to systems engineering the MFD is a system of systems (SoS) [9]. Therefore the embedded character has to be taken into account, requiring “systems thinking” [10].

A current method for developing embedded systems is the “3 Phasen V-Modell” of K. Bender [11], a modified V-model [12] for embedded systems. In this model the system is partitioned early into its (sub-)systems and disciplines. The (sub-)systems, to which the MFD and the dust filter belong to, should be designed independently among themselves [11] providing parallel, effective and time-saving development. For a (sub-)system that is highly dependent from others like the MFD, this concept of early partition and independent development is too vague when it comes to how information of one (sub-)system is linked to the other (sub-)system. Looking at the use case of the new dust filter installation the system requirement “new dust filter and display its state on the MFD” has to be broken down to the subsystem level of the dust filter - “new dust filter” - and the MFD - “add a new filter symbol”. From that point on the requirement is to be refined to the MFD component level where the SW specification is modified. The sequential requirement process of analysis, breaking down and freezing is non-qualified for frequent changes.

III. HOLISTIC CONCEPT FOR EMBEDDED MFD

This article presents a holistic concept for improving the SW maintenance of embedded MFDs. This concept combines the “3 Phasen V-Modell” with the basic ideas of display SW development according to CRF and MDA. The concept consists of five levels Domain, General MFD, Domain Specific MFD, Platform and Implementation classified according to the levels of MDA [13]. The focus of this article is to present the model of the domain level and its theoretical effect on the SW maintenance process. Fig. 2 shows an overview about the concept.

Fig. 2. Holistic concept for embedded MFDs integrated in the “3 phasen V-Modell” of K. Bender.

Previous work [14] outlined the MFD related issues for improving SW maintenance however this article will show an approach addressing the SW maintenance challenges at system level. Thus the emphasis is the first level “0. Domain” being outlined more detail than the other levels.

A. Domain

This specification belongs to the system level providing information of the system design in form of a product structure. Here the all subsystems have to be categorized; their state and functions are to be defined. Already at this level display relevant information of all subsystems has to be referred to the MFD. Since this embedded MFD concept is not limited to a specific domain the implementation of the appropriate product structure is the choice of the end user. Fig. 3 shows the intermediate model describing the reference of (sub-)systems to the MFD.

The general aspect of a product structure is to provide the architectural relationship [15] derived from the system specification. The generalization titled “consists of” provides the refining of “System Functional Specification” into “System Internal Functions”, “System External Functions”
and “Display Functions”. The association “display” indicates that subsystems need the MFD for displaying their states. Although MFD is an embedded system, the separation of “Embedded MFD Subsystem” from “Embedded Subsystem” is on purpose. Appropriate to the “3-Phasen V-Modell”, every subsystem consists of mechanic, HW and SW components.

![Fig. 3. Class diagram presenting the reference of subsystems to MFD.](image)

![Fig. 4. General MFD model.](image)

The association “display” indicates that subsystems need the MFD for displaying their states. Although MFD is an “Embedded MFD Subsystem” the separation of it from “Embedded Subsystem” is on purpose. The most important feature of the class diagram is the possibility to communicate display relevant changes of subsystems to the MFD. Display relevant changes are defined in the comment (Fig. 3). The monitoring of those changes by the MFD is implemented with the design pattern “Listener”. The interface “Display Change Information” adds or removes relevant Listener who is interested in display relevant changes. Those listeners and their actions in reply to the display changes are defined in the interface “MFD Change Request” and the method “MFDChangeRequest()”. The implementing class of this interface is “Embedded MFD Subsystem”. This model belongs to the Computer Independent Model (CIM) according to MDA.

**B. General MFD**

The general MFD specifications belong to the subsystem level containing all information about domain independent functional and non-functional MFD requirements, e.g. the display functions of system information or reacting to user and system inputs. Furthermore MFD design decisions are defined. The embedded MFD concept requires the specification of the graphical representation, thus the look
and feel of symbols, to be separated from the logic design, triggering the symbols during MFD operation based on [2] and [16]. Since the change of the logic leads to change in the logic this feature is also implemented using the pattern design “Listener”. Fig. 4 just provides one intermediate model of several in this level defining the general MFD. The models defining the general MFD architecture also belong to the Platform Independent Model (PIM) according to MDA.

C. Domain Specific MFD Specification

The domain specific MFD specifications adapt general MFD specifications with display relevant information from the system, defined in the domain level (see bent arrow from 0. Domain to 2. Domain Specific MFD in Fig. 2). All general MFD models can be reused here enriched with domain specific information from the domain level denoted by figure 5. The models here also belong to the PIM level according to MDA. Using the example of installing a new filter (sub-)system ideally the updated MFD layout with the filter could be generated when “Domain Specific Information” (Fig. 5) are given. These could be domain specific standards, interface connections, specification of the filter system (such as range, critical values etc.) and the location of the new filter system within the cockpit. With this information the filter symbol could be parameterized and integrated in the existing MFD layout.

D. Platform Specification and Implementation

Here the HW and SW Platforms of the operating MFD are described (Platform Model according to MDA) for defining transformation rules of code generators. Due to the MDA approach this step automatically derived from the domain specific MFD specifications and the platform descriptions.

IV. CONCLUSIONS AND OUTLOOK

The early reference of display relevant (sub-)systems to the MFD provides a structured process of when information is delivered to the MFD facilitating current processes in the practice. In case of a display relevant (sub-)system change, the MFD will be informed. This holistic view is particularly important for the MFD SW maintenance. The findings presented in this article show that embedded MFDs are more often affected by modifications due system run-time than other (sub-)systems. Furthermore embedded MFDs cannot be treated as a common (sub-)system because of their the high dependency on others. This issue is not picked out as a central theme in the literature yet. This article provides an approach to how to diminish embedded MFD maintenance challenges according current methods by observing the problems at system level. Developing the concept according to the MDA approach because of its influence on the System Design, the concept would have the potential to automate MFD maintenance processes partly. This concept is developed based on the daily MFD SW maintenance challenges in the industry. At this state of research only the theoretical foundations are provided.

For the next steps, it is intended to focus research on defining mapping rules for model and code transformations and to adopt the concept in the industrial environment.

REFERENCES


Qi Zhang was born in China 1986. She studied electric engineering and graduated with a diploma (equivalent to a M.Sc.) in 2009.

Actually she works as a PhD student at the Economic Informatics Institute at the University of the Federal Armed Forces Munich in cooperation with the system support center of Eurocopter Germany for NH90/Tiger.

Her primary interest is to develop a MDA compliant development concept for multi-function display systems in order to optimize the SW maintenance. Main research fields are SW Engineering, model driven SW development and Systems Engineering.

Andreas Karcher studied computer science and graduated with a diploma (equivalent to a M.Sc.) in 1986. After 5 years of working as a project director for an IT-consultancy in Karlsruhe, 1992 he joined the Technische Universität München and received his doctoral degree there in 1996. Since February 2004 Prof. Karcher holds a tenure track University Professorship for Application Integration at the Institute of Applied Informatics at the University of the Federal Armed Forces in Munich, Germany. His main research topics are Middleware and Information Integration, IT Services and Enterprise Architectures, mobile Cloud-based Application Integration, System and Product Lifecycle Management, Embedded Quality for multi-function display systems in order to optimize the SW maintenance.