Proc. of the 39th Annual Conference of the IEEE Industrial Electronics Society, pages 4366–4371, 2013

Design Patterns for Separating Fault Handling from Control Code in Discrete Manufacturing Systems

Michael Steinegger*, Alois Zoitl**, Martin Fein*, and Georg Schitter*

*Automation and Control Institute, Vienna University of Technology
Gußhausstraße 27-29, AT-1040 Vienna, Austria
Email: {steinegger,schitter}@acin.tuwien.ac.at

**Fortiss GmbH, Guerickestraße 25
DE-80805 Munich, Germany
Email: zoitl@fortiss.de

***Abstract*—The objective of this paper is to propose a methodology for strictly separating fault detection and fault handling methods from control code in discrete manufacturing systems. In order to enable the separation, two different design patterns are elaborated. Based on an analysis of typical faults in manufacturing systems, a library of reusable function blocks for fault detection, handling, and also fault recovery is developed. These function blocks are implemented according to the international standard IEC 61499 and are applied within the design patterns. The design patterns are evaluated with the sequential control of an injection molding machine. It is shown, that the design patterns simplify the engineering and the separation of the fault detection and fault handling from control code significantly, since they provide clear design rules. Applying the proposed design patterns reduces the complexity of industrial control applications drastically. Furthermore, the function block library provides reusable and proven fault detection and fault handling methods.**

I. Introduction

Requirements on modern manufacturing systems increase steadily due to international competition and resulting cost pressure. The manufacturing systems as well as their associated control applications become more and more complex since they should be able to execute several complex tasks. This also results in increasing demands on safety and security. The plant itself should be protected and hazardous situations for humans and the environment have to be avoided. To fulfill the safety requirements, system faults (e.g., caused by component failures) have to be detected and handled at an early stage.

Today, the vast majority of code pieces of typical industrial control applications is focused on the safe response to plant misbehavior (e.g., realized as safety interlocks) rather than on controlling the functional behavior of the plant. For example, fault handling in robotic cells takes up to 80% of the complete control application, as observed by Smith and Gini [1]. Current evaluations of typical manufacturing control applications also showed that the average amount of control code for normal operations without fault handling is about 17% (cf. Güttel [2]).

Increasing complexity and safety requirements for manufacturing systems result in complex and often hard to understand and maintain control applications, since the fault detection and fault handling methods are usually directly implemented and intertwined with the control code. One of the reasons is the huge effort to strictly separate and decouple control code from fault detection and fault handling methods due to missing clear design guidelines. As a consequence, subsequent changes in the control applications becomes challenging, error-prone, and often increase the complexity of the application again.

In almost all engineering disciplines, design patterns are elaborated to provide optimized and reusable templates which can be applied to a wide range of related problems. For industrial automation there exist several design patterns for flexible manufacturing based on the standard ISA-88 (cf. Brandl [3]) or distributed control systems like the MVCDA (Model-View-Controller-Diagnostic-Adapter) design patterns proposed by Christensen [4]. Lee and Tilbury [5] developed a modular controller design methodology for manufacturing systems, which also considers integrated fault handling on manufacturing cell level. Furthermore, Ferrarini et al. [6] proposed a methodology for fault isolation in control systems based on *programmable logic controller*s (PLCs). The presented approach also opens up new possibilities to separate control for normal operations from fault detection in the PLC-based controller design [7]. This separation is achieved within the control model, where the non-nominal behavior (system faults) represents a sub-model.

Serna et al. [8] elaborated two design patterns for failure management in control applications, implemented according to the international standard IEC 61499 [9]. Therein, each *Function Block* (FB) is attached to a so-called failure zone. These failure zones allow logical grouping and definition of hierarchies of FBs. The second pattern is designed for failure supervision. This pattern allows to detect errors which cannot be detected by a FB on its own, for example, an error which only occurs if a group of FBs are in a specific state. However, each FB in this framework has its own failure detection.

Currently, there exist no clear design rules for a strict separation and decoupling of control code for normal operations from fault detection and fault handling methods. Inspired by several existing approaches, enabling the mentioned separation for exception handling within workflow management systems [10]–[13], two different design patterns are proposed for discrete manufacturing systems. In contrast to [8], this paper is focused on decoupling the usually tightly coupled and intertwined control code and fault detection and fault handling methods.

The paper is organized as follows. In Section II, the methodology of the design patterns definition is stated. Based on the proposed design patterns, the implementation according to the international standard IEC 61499 is explained in Section III. The design patterns and their implementations are evaluated by applying them to an injection molding machine, as described in Section IV. Finally, Section V concludes the paper.

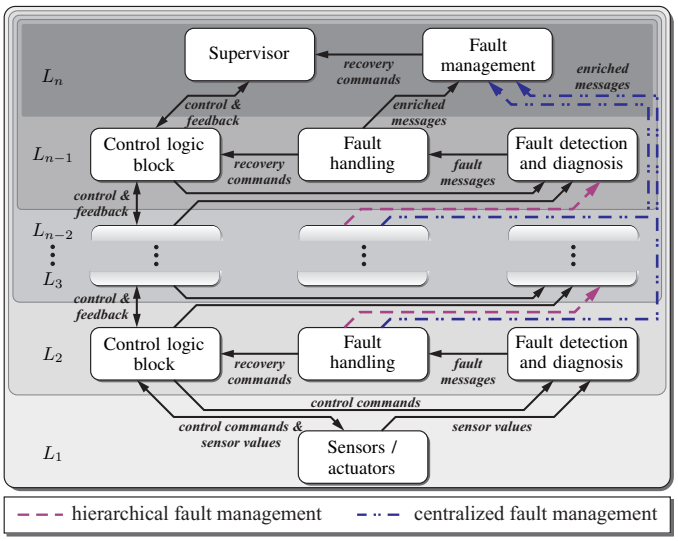

Fig. 1. Two different design patterns for fault detection, fault handling, and fault management in hierarchically structured manufacturing systems.

II. Concept

In this section, the design methodology and two resulting design patterns for strictly separating and decouple control code for normal operations from fault detection and fault handling methods in manufacturing systems are described. Throughout the paper, the terminology for fault detection, diagnosis, and handling according to Isermann and Ballé [14] is applied.

A. Hierarchical Composition of Control Systems

The mechanical construction of modern manufacturing systems is often based on a modular structure of simple components which are composed to achieve more complex architectures (cf. Sünder et al. [15]). Hierarchical composition not only applies to physical structures. Industrial control applications are also often structured in a component-based and hierarchical manner to cope with the increasing complexity of manufacturing systems and, subsequently, the control software itself.

Typical hierarchical control software design for a hierarchy $H = \{L_n, L_{n-1}, \ldots, L_1\}$, where n is the number of levels, applies a supervisory control block on the highest hierarchy level L_n. This supervisory block takes care of the control of the overall system, coordinates control functions on the lower control levels, and is in charge of reacting to unexpected system misbehavior. It sends control commands and receives specific feedback from subsequent control levels $L_i, i \in \{2, \ldots, n-1\}$. The subordinate components on the control level L_{n-1} coordinate the components on the next lower level and, thus, propagate the commands from the supervisory level to the subordinate components they are responsible for. Furthermore, they propagate and comprise the feedback from the lower level to the next upper level. The hardware-near level L_1 represents the interface to the sensors and actuators. As a consequence, the supervisory level cannot directly access the hardware-near level since there is always at least one level in between, containing components to directly control the hardware.

B. Design Methodology

In order to define design patterns for separating and decouple control code for normal operations from fault detection and fault handling methods applied to hierarchical manufacturing control structures, the following engineering steps are executed.

First, a fault analysis is conducted in order to identify possible errors and faults in the system under consideration. Established methods like FMEA (Failure Mode and Effects Analysis) or FTA (Fault Tree Analysis) are suitable for that task, since for example FTA provides a simple possibility to represent faults and their root causes in a graphical way.

Based on the above mentioned fault analysis, common faults and their root causes are identified. This enables the implementation of standardized and proven methods for detecting and handling typical faults in discrete manufacturing systems.

C. Design Pattern with hierarchical Fault Management

Based on the proposed engineering methodology presented in the previous section, two different design patterns for decoupling control from fault detection code are depicted in Fig. 1. Both design patterns take hierarchically composed structures into account, as described in Section II-A. As a result, the highest level represents the supervisory level L_n, the lowest is the hardware-near level L_1, and L_i are the $n-2$ control levels.

The fault management of the first design pattern is realized in a central block on the supervisory level L_n. In contrast, the second design pattern applies a hierarchical fault management such that each level in a hierarchical control structure has its own fault management incorporated into fault handling blocks.

In the following subsections, the required building blocks for fault detection, handling, and management of the design pattern are described. Furthermore, the interaction of these building blocks to establish a hierarchical fault management is stated.

1) Fault Detection: Each control level L_i incorporates a set of fault detection blocks in order to detect faults on the subordinated level based on standardized and encapsulated fault detection methods. For a reliable fault detection and also fault handling, the feedback from the subordinated level, the actual sensor values, propagated upwards from the lower control or hardware-near blocks, as well as the control values from the control logic blocks on the same control level have to be provided to the fault detection and diagnosis FBs. If one of the fault detection methods detects one or more faults, fault messages are generated by the diagnosis block. These fault messages encode the occurred fault type, time stamp and other additional information and are transferred from the fault detection and diagnosis block to the fault handling block.

2) Fault Handling: Based on the received fault message, the fault handling FB executes appropriate methods in order to remove the occurred faults. Therefore, it directly influences control logic FBs on the same hierarchy level. The fault handling FB provides methods for eliminating faults or, if an elimination was not successful, for graceful degradation, restarting (or start redundant) components or even to stop components directly. However, these methods only affect the control logic FBs on the same hierarchy level and the controlled hardware component. The received fault message is enriched with additional information about the handling status and is then sent to the fault detection and diagnosis component of the superior level in the control hierarchy. It should be noted that in the hierarchical fault management approach, the fault messages are further enriched with fault information from each fault detection and diagnosis FB, if an fault occurred on the subordinate level. The advantage of this approach is that hard to detect faults, which are not detected by FBs on a single

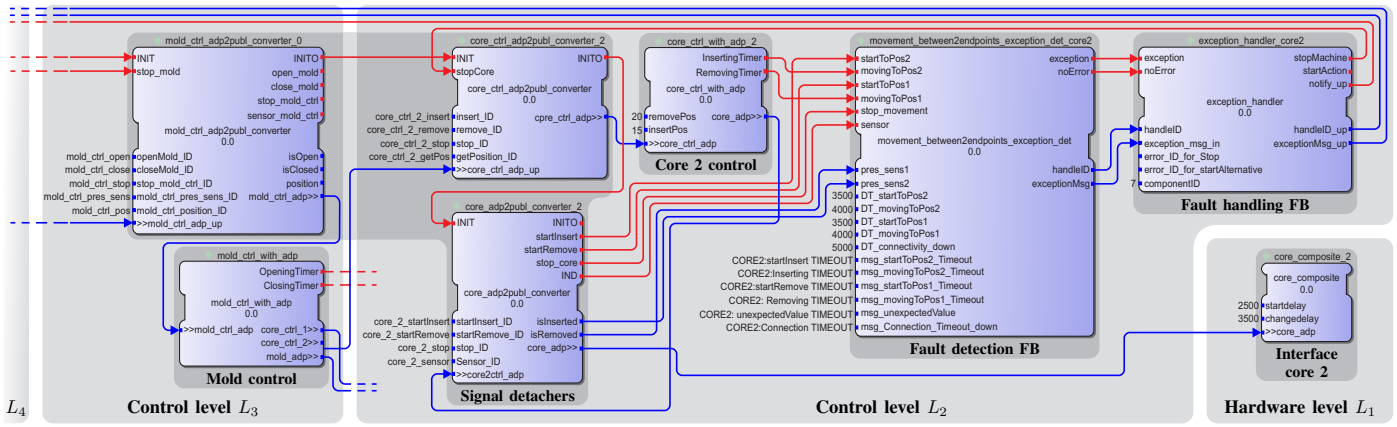

Fig. 2. Implementation snippet representing the mold control and the control of the second core according to the international standard IEC 61499 in 4DIAC. The implementation is based on the proposed centralized design pattern, where the data connections (blue lines) and event connections (red lines) of the fault handling function block (FB) are connected to the centralized fault management on the supervisory level L_4. For simplicity reasons, the presented implementation example only contains a single fault detection FB and fault handling FB instead of a subapplication with grouped fault detection and fault handling FBs.

hierarchy level, can be detected by superior levels, since they have access to more data and information about the actual system state. The enriched fault messages are propagated to the fault management block. In the hierarchical fault management approach, the fault management is part of fault handling FBs.

3) Fault Management: The fault management block manages all fault messages and provides methods which have, in contrast to the fault handling methods, effect on the superior level. For example, if there exists a faulty component in the system, other faultless components involved into the same process as the faulty component can be stopped in order to prevent them from possible damage. On the supervisory level L_n, the fault management block is further responsible for logging of occurred faults and preparation for visualization.

D. Design Pattern with centralized Fault Management

The design pattern presented in the previous section possesses fault management blocks on each hierarchy level. However, it is also possible to centralize the fault management on the supervisory level L_n. Then each fault handling block on the subordinate hierarchy levels propagates its fault messages directly to the fault management block on the highest level. As a consequence, the fault handling blocks do not directly receive any fault messages from the subordinate levels.

The advantage of this approach is that the fault messages can be faster processed since they are not propagated through all hierarchy levels. Furthermore, the fault handling blocks are easier to implement because the fault management only occurs on the level L_n. However, the disadvantage is that in case of a fault in the supervisory level, the complete fault management may fail due to its centralized structure.

III. Implementation

In this section, the implementation of the proposed design patterns according to the international standard IEC 61499 is described. Here, the focus lies on the possibilities for a strict separation and decoupling of control code from fault detection and fault handling methods in IEC 61499 control applications. The FB concept, where FBs are seen as software components being applied separately from each other without knowing about their internal function (cf. Sünder et al. [16]), is suitable for that. Since each FB has event and data inputs or outputs, representing the interface, the adapter interface concept proposed in the international standard IEC 61499 is applied to reduce the number of connections between the FBs. Furthermore, Zoitl and Prähofer [17] evaluated the IEC 61499 modeling language as suitable for arranging FBs in a hierarchical manner.

A. Design Pattern Implementation according to IEC 61499

To meet the high demands on modularity and interchangeability for component-based hierarchical automation solutions, the implementation of the design pattern with centralized fault management, which is described in Section II-D, is based on the international standard IEC 61499. The main building block for hierarchical decomposition is the FB with its inputs and outputs for events and data. Complex industrial control applications often require a high amount of data and event signals which are exchanged between the different hierarchy levels. This results in complex and often hard to understand and maintain FB networks due to a cluttered design space.

To reduce the complexity, adapter interfaces are applied as interfaces between the corresponding hierarchy levels. An adapter interface combines all bidirectional event and data connections to one common interface specification. Each control level L_i consists of at least one control logic FB, which controls the lower components via an adapter interface. The control logic FBs are extended by a fault detection and diagnosis FB, as well as a fault handling FB. Signal detachers for adapters are necessary to provide the required event and data information to the fault detection methods. On the one hand signal detachers forward the adapter signals through the FB, and on the other hand event and data signals are sent to fault detection FBs.

An example for the implementation of the design patterns according to the international standard IEC 61499 is illustrated in Fig. 2. The control level of the second core of an injection molding machine (see Fig. 3) is illustrated in detail, where the total number of hierarchy levels is $n = 4$. In the presented example, faults occurred during movements of one of the mold cores are recognized by a fault detection FB which generates fault-specific messages. The type of the occurred fault is encoded and provided at the `handleID` data output of the fault detection FB. Furthermore, the fault-specific mes-

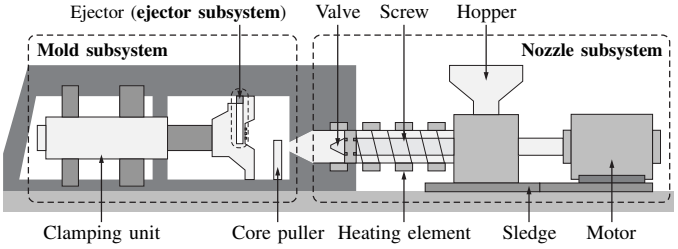

Fig. 3. Schematic construction of the injection molding machine used for the evaluation study (based on Prähofer et al. [18]).

sages, generated by the fault detection FB, are enriched with additional fault information and sent via the `exceptionMSg` port to the fault handling FB. The fault handling FB decides which strategy (e.g., start new movement attempt) is applied in case of an occurred fault. It directly influences the control logic FB by starting individual methods for fault elimination, which is stopping the core movement immediately via the `stopMachine` output. In this case, the functional interaction of the fault handling FB and control logic FB can be seen as a special kind of override control realization. Furthermore, the fault handling FB adds specific component information to the fault message received from the fault detection FB and sends it to the fault management FB on the highest hierarchy level.

The centralized fault management FB is located on the supervisory level L_4 and collects all fault messages sent from the fault detection and handling FBs. As a result, this FB has detailed knowledge of the actual fault states of the overall system being controlled. In case of a fault this FB effects the overall manufacturing system by stopping the overall system or transfers the system to a safe state. Apart from that, the fault management FB provides a filtered error log for users.

B. Interaction with Control Logic Function Blocks

The overall goal of the presented approach is to strictly separate and decouple normal operation control code from fault detection and fault handling FBs. Typically, the fault detection code is in charge of supervising lower level components and identify whether they are behaving as expected. In order to achieve this goal, the fault detection FBs require information on the commands sent by the control logic FBs to the lower level components (e.g., to move an axis) and on the feedback provided by the lower level component (e.g., completion of axis motion). In the proposed approach so-called signal detacher blocks are added between the control blocks and the lower level components. These signal detachers are in charge of acquiring the commands to and the feedback from the lower level components and forward them to the fault detection FBs on the same hierarchy level. Several possibilities to implement the signal detacher FBs according to the standard IEC 61499 are available, which are discussed in the following subsections. It should be noted that connections between FBs can be generated (semi-)automatically by applying model-based approaches.

1) Local Multicast: In order to decouple IEC 61499 application parts, Christensen [4] introduced the local multicast design pattern. The idea behind the local multicast approach is to establish a software bus within a device and to allow topic-based data exchange between application parts similar to the publish/subscribe communication model in distributed systems. In topic-based data exchange, a unique identifier is used to address and identify the sent data and can be utilized by subscriber FBs to receive data from the publishing FB.

The advantage of applying the local multicast approach in the signal detacher FBs is that they are completely separated from the fault detection FBs. Thus, the fault handling could easily be implemented or maintained and changed independently from the signal detacher. The assignment of unique identifiers for different communication channels in the local multicast approach, however, represents a disadvantage regarding re-usability. In case of reusing the same signal detacher FB and fault detection FB at different locations within the control application (e.g., for repetitive structures), the unique identifiers have to be adjusted manually. This, however, reduces the advantages of re-usability of the developed fault detection FBs.

2) Direct Connection: Another possible approach is to use normal IEC 61499 connections (i.e., event, data, and adapter connections) for directly connecting the signal detacher with the fault detection FBs. The drawback of this approach, especially for supervising complicated lower level components with many events and data exchanged in both directions, is that often many direct connections have to be established between signal detacher and the fault detection FBs. This may result in a cluttered design space, making it hard to understand and maintain the implemented control applications. However, no additional work is necessary in case of reusing the signal detacher or fault detection FBs in several different locations.

3) Common Subapplication: Applying the subapplication concept, defined in the international standard IEC 61499, helps to overcome the issues of cluttered design spaces with direct connections, as discussed in the previous subsection. Subapplications are a means for structuring control applications in IEC 61499 by grouping application parts together. The grouped application elements can, similar to *Composite Function Block*s (CFBs), be saved as types and reused in the same or even a different control application. In difference to CFBs the content of subapplications can be modified within applications.

The control application, implemented for evaluation of the presented approach, is realized with fault handling subapplication. This subapplication contains the signal detacher and all FBs relevant for fault detection and fault handling. Furthermore, the fault handling subapplication is established between the control FB and the lower-level components. The advantage of this approach is that the overall application layout is clearly structured and the fault handling subapplication can be implemented completely independent from the control code.

C. Function Block Library for Fault Detection

The implementation described in the previous sections provides a framework for an efficient way to implement fault detection and fault handling in industrial automation systems. In order to further reduce the implementation effort, common faults in discrete manufacturing automation systems are investigated, that need to be detected and handled. This research reveals a set of common faults, which are: signal timeout (e.g., waiting for the rising edge of a sensor indicating the end of a movement), connection timeout, detection of wrong states, unexpected values, range expiration, or gradient expiration. For each of these faults or fault classes an own fault detection FB has been implemented. Components of the implemented FBs library can be directly applied for fault detection and can also, dependent on their parametrization, be combined with each other in order to detect also normally hard to identify faults.

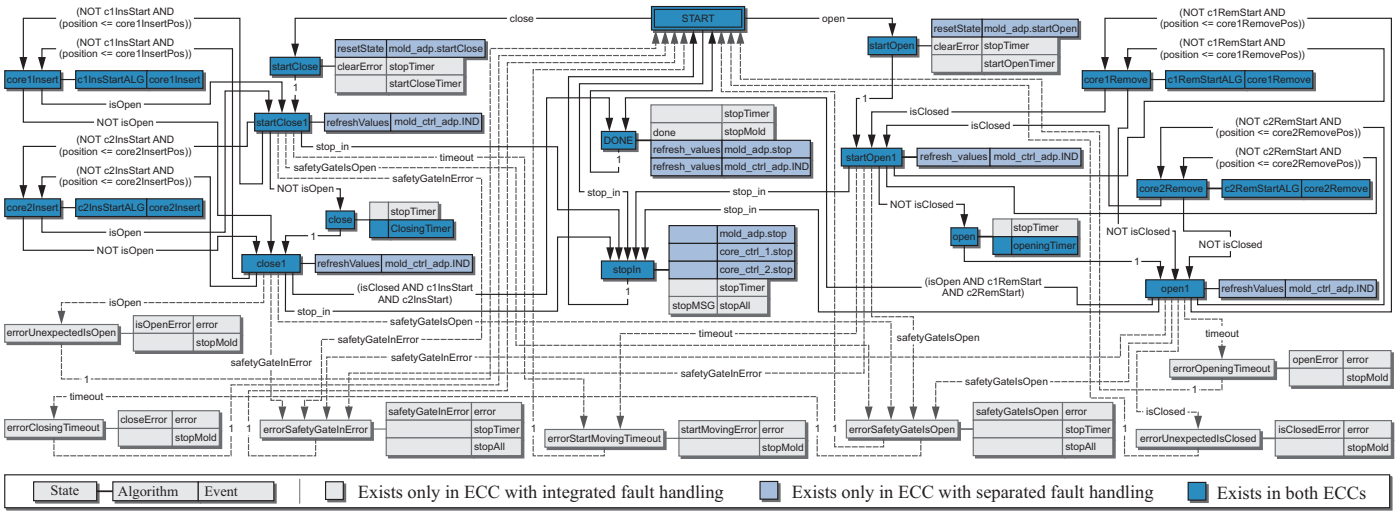

Fig. 5. Graphical comparison of the execution control charts (ECC), representing the mold control, for integrated and strictly separated fault detection and fault handling. Applying one of the proposed design patterns significantly increases the readability and maintainability of the ECC since all fault states (light gray) and their connections (gray dashed lines) are outsourced to standardized function blocks.

IV. Evaluation

In order to evaluate the proposed design patterns, two different control applications for an injection molding machine have been implemented. One control application was realized with directly integrated fault detection and fault handling methods into the control FBs, as it is usually realized in existing industrial implementations. The second implementation is based on the strict separation and decoupling of the control code from fault detection and fault handling methods. It is implemented according to the design pattern with centralized fault management on the supervisory level, as described in Section II-D.

A. Injection Molding Machine

In industrial manufacturing, injection molding machines are applied to produce costumer-specific plastic parts. According to Fig. 3, the injection molding machine applied for evaluation is divided into three subsystems. The nozzle subsystem constitutes of a heating element and a screw, where the raw material is heated up and injected by the screw under pressure into the mold subsystem. The mold subsystem contains a mold as well as two cores, which can be exchanged to vary the form of the produced plastic parts. After injection of the heated raw material into the mold pattern and a short cool-down phase, the mold is opened and the produced part is ejected by an ejector, representing the third subsystem of the molding machine.

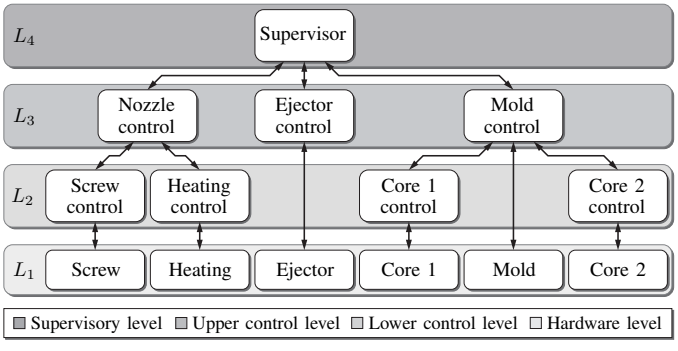

Fig. 4. Hierarchical control structure of the injection molding machine. Fault detection and fault handling blocks are neglected due to readability reasons.

Figure 4 depicts the control structure of the injection molding machine, representing a control hierarchy with $n = 4$ levels. The FBs on the lowest level L_1 act as interfaces to the hardware and thus directly communicate with the sensors and actuators. These FBs transform primitive functions into elemental control program units and can also act as simulation blocks in case of not physically existing hardware. Level L_2 consists of the units which are in charge of controlling the components of the modular-structured subsystems like the nozzle or mold subsystem. Control FBs on the highest control level L_3 manage these modular-structured subsystems or directly control the elementary subsystems like the ejector. The supervisor on the highest hierarchy level L_4 coordinates all control FBs on the subordinate control levels. It further supports several modes of operation, like fully-automatic or semi-automatic operation.

B. Evaluation of the proposed Design Patterns

The hierarchically structured control software for the injection molding machine has been implemented according to the international standard IEC 61499 in 4DIAC[1], as described in Section III-A. All hierarchy levels were connected via adapter interfaces, represented as bidirectional arrows in Fig. 4.

As a representative example with medium level complexity, the ECCs of the mold control for both implemented control applications are illustrated in Fig. 5. The ECCs represent the logical control structure, e.g., for opening (transient states on the right-hand side in Fig. 5) or closing the mold (transient states on the left-hand side). Furthermore, they incorporate the final states (e.g., system stopped or molding cycle done) and, in case of integrated fault detection and fault handling, the states which are reached due to occurring faults in the system. It can be seen that the complexity of the ECC decreases drastically if one of the proposed design patterns is applied to the control software design. Since the fault detection

[1]4DIAC homepage: `http://www.fordiac.org/`

TABLE I. COMPARISON BETWEEN INTEGRATED AND STRICTLY SEPARATED FAULT DETECTION AND FAULT HANDLING METHODS.

Control component	**Fault detection & fault handling**			
	Integrated		Separated	
	States	*Transitions*	*States*	*Transitions*
Control level L_3				
Nozzle	6	12	6 (±0%)	15 (+25%)
Ejector	17	37	7 (-59%)	17 (-54%)
Mold	22	54	15 (-32%)	38 (-30%)
Control level L_2				
Screw	14	26	7 (-50%)	15 (-42%)
Heating	8	13	7 (-13%)	12 (-8%)
Core 1	14	26	7 (-50%)	15 (-42%)
Core 2	14	26	7 (-50%)	15 (-42%)
Total reduction	95	194	56 (-41%)	127 (-34.5%)

states are encapsulated into standardized fault detection and handling blocks, the overall number of states and transitions do not decrease significantly. However, the understandability and maintainability of the control code considerably increases due to the strictly separated and decoupled code structures.

C. Discussion

The comparison between integrated and separated fault detection and fault handling is summarized in Table I. It can be seen that the number of states and transitions within the control blocks of the hierarchy levels L_2 and L_3 for the whole injection molding machine is reduced by a total average of 36.7%. Beside the significant complexity reduction of the control FBs, a further big advantage of the proposed design patterns are the elaborated FBs for fault detection and fault handling. Since faults and their root causes are systematically analyzed, fault detection and fault handling methods for common faults can be standardized and encapsulated within fault detection FBs. From an engineering point of view this is beneficial since these FBs represent encapsulated, tested, and proven software components which can be reused in further projects without implementing the fault detection and fault handling from scratch. Furthermore, the elaborated FB library helps to reduce engineering time and a strict application of one of the proposed design patterns significantly enhances the overall quality of the implemented industrial control applications.

V. CONCLUSION

This paper presents a design methodology to strictly separate and decouple control code for normal operations from fault detection and fault handling methods in discrete manufacturing systems. Both resulting design patterns take hierarchical control software design into account and realize a centralized as well as a hierarchically-structured fault management. The design patterns are implemented according to the international standard IEC 61499, wherefore several possibilities to realize a strict separation and decoupling of the code parts are discussed. Fault detection and fault handling within the implemented design patterns is achieved by applying standardized FBs from an implemented FB library. These library FBs make it possible to detect, diagnose, and handle typical faults or even component failures in manufacturing systems, which were identified during an extensive analysis of several different automation systems.

The proposed design methodology as well as the implemented FB library was applied to an injection molding machine for evaluation. It is shown that the design patterns scale up very well due to their hierarchical structure and their strict design guidelines. The application of the proposed design patterns for a strict separation and decoupling of control code from fault detection and fault handling showed a state reduction by in total 41% and a total transition reduction by 34.5% within an ECC of a typical manufacturing control implementation. Both design patterns help, beside enhancing the overall code quality and increasing maintainability, also to save time during the engineering process of industrial control applications.

REFERENCES

[1] R. E. Smith and M. Gini, "Reliable realtime robot operation employing intelligent forward recovery," *Journal of Robotic Systems*, vol. 3, no. 3, pp. 281–300, 1986.

[2] K. Güttel, *Konzept zur Generierung von Steuerungscode für Fertigungsanlagen unter Verwendung wissensbasierter Methoden [german]*, ser. VDI Fortschritt-Berichte, A. Fay and K. Krüger, Eds. VDI Verlag, 2013, vol. 444.

[3] D. Brandl, *Design Patterns for flexible Manufacturing*. International Society of Automation (ISA), 2006.

[4] J. Christensen, "Design patterns for systems engineering in IEC 61499," in *Fachtagung Verteilte Automatisierung*, 2000, pp. 269–276.

[5] S. Lee and D. Tilbury, "A modular control design method for a flexible manufacturing cell including error handling," *International Journal of Flexible Manufacturing Systems*, vol. 19, no. 3, pp. 308–330, 2007.

[6] L. Ferrarini, M. Allevi, and A. Dedé, "A methodology for fault isolation and identification in automated equipments," in *Proc. of the 9th IEEE Int. Conf. on Industrial Informatics*, 2011, pp. 157–162.

[7] L. Ferrarini, M. Allevi, and A. Dedé, "Design and implementation of an automatic on-line diagnosis with TiDiaM and TiDE," in *Proc. of the 15th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2010, pp. 1–6.

[8] F. Serna, C. Catalán, A. Blesa, and J. Rams, "Design patterns for failure management in IEC 61499 function blocks," in *Proc. of the 15th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2010, pp. 1–7.

[9] Int. Electrotechnical Commision, "IEC 61499 – Function blocks, Part 1: Architecture," January 2005.

[10] Z. Luo, A. Sheth, K. Kochut, and J. Miller, "Exception handling in workflow systems," *Applied Intelligence*, vol. 13, pp. 125–147, 2000.

[11] C. Hagen and G. Alonso, "Exception handling in workflow management systems," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp. 943–958, 2000.

[12] J. Li, Y. Mai, and G. Butler, "Implementing exception handling policies for workflow management system," in *10th Asia-Pacific Software Engineering Conference*, 2003, pp. 564–573.

[13] N. Russell, W. Aalst, and A. Hofstede, "Workflow exception patterns," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, E. Dubois and K. Pohl, Eds. Springer Berlin Heidelberg, 2006, vol. 4001, pp. 288–302.

[14] R. Isermann and P. Ballé, "Trends in the application of model-based fault detection and diagnosis of technical processes," *Control Engineering Practice*, vol. 5, no. 5, pp. 709–719, 1997.

[15] C. Sünder, A. Zoitl, M. Rainbauer, and B. Favre-Bulle, "Hierarchical control modelling architecture for modular distributed automation systems," in *IEEE Int. Conf. on Industrial Informatics*, 2006, pp. 12–17.

[16] C. Sünder, A. Zoitl, J. Christensen, H. Steininger, and J. Fritsche, "Considering IEC 61131-3 and IEC 61499 in the context of component frameworks," in *Proc. of the 6th IEEE Int. Conf. on Industrial Informatics*, 2008, pp. 277–282.

[17] A. Zoitl and H. Prähofer, "Building hierarchical automation solutions in the IEC 61499 modeling language," in *Proc. of the 9th IEEE Int. Conf. on Industrial Informatics*, 2011, pp. 557–564.

[18] H. Prähofer, D. Hurnaus, R. Schatz, C. Wirth, and H. Mössenböck, "Monaco: A DSL approach for programming automation systems," in *Conference on Software Engineering*, 2008, pp. 242–256.