Defining IEC 61499 Compliance Profiles using UML and OCL

Tanvir Hussain and Georg Frey, Member, IEEE

Abstract—IEC 61499 holds its distinct identity as a standard modeling language for distributed control applications in the realm of Industrial Process Measurement and Control Systems (IPMCSs). From the perspective of modeling distributed control applications it seems to be a fitting standard unless one tries to map the models into implementations on any of the diverse hardware platforms available for IPMCS. UML often seems to be a modeling counterpart of IEC 61499. Due to its wide-spread application in different domains UML is now enriched enough with artifacts like standard profiles and hence more expressive. Moreover, its familiarity inspired researchers to draw modeling similes with IEC 61499. In this context the presented work tries to present the aspects of UML artifacts to specify IEC 61499 models to avoid certain ambiguities that might widen the gap between model and implementation. Moreover, considering the fact that the model of IEC 61499 resources and its mapping to implementation determines how the constituent element should behave in the application, a modeling paradigm for resources – taking into account the execution semantics (profile) – in terms of UML is also proposed. Its uniqueness lies in the fact that the ambiguity of textual representation is sufficiently avoided. Furthermore, the definition in terms of constraints (OCL) allows checking a model for consistency with a given profile.

I. INTRODUCTION

Development of distributed control applications for Industrial Process Measurement and Control Systems (IPMCSs) is growing to be a challenging task. The advent of improved hardware facilities is on one hand enabling the fulfillment of tight hard and soft real time needs and on the other hand inspiring to build up a heterogeneous platform for the control applications that should be deployed on these. Middleware systems are often taking care of the interoperability making the development of control software running on top of them to be easier. Due to the numerous available choices of hardware platforms and networking technologies for industrial applications, middleware systems applicable for various combinations of hardware and networking platforms are coming into being more frequently than ever. Moreover, short life cycles of hardware are often inducing major or minor modifications to the middleware. Apart from this, certain industries also undergo changes due to change of products and production technologies, the frequency of which might vary from weeks to years.

With the promise of helping in coping up with these challenges through offering flexibility and reconfigurability IEC 61499 was introduced [1], [2]. It has to be noted that it is a modeling language and therefore does not precisely define the details of implementation. Even in the latest version, the standard does not clearly specify the implementation guidelines for the defined modeling artifacts. This appears to be a source of ambiguity as a modeled entity might have different interpretations at the analysis and implementation phases of development.

In the absence of a strict guideline for transformation from modeling artifacts of IEC 61499 to the corresponding executable codes, certain consistent guidelines or interpretations of the standards are often applied by its users. Quite often the interpretations and the related assumptions differ from one user of the standard to another [3]. The difference of interpretations might be resolved if the guidelines are specified properly in the form of a compliance profile which might give some concrete idea about which allowable variation of the interpretation of the IEC 61499 models are taken while analyzing the models or implementing corresponding codes. This can be done in any of the three forms, namely,

1. Textual specification
2. Formal Discrete Event System (DES) model
3. Unified Modeling Language (UML)

In this paper it is proposed that a combination of 1 and 3 could be a reasonable choice. This is because UML is now quite popular as a modeling language and the visual representations give a better overview of the system and could be more comprehensive than textual expressions alone. Moreover, it is possible to make use of UML profiles which have shown to be useful in describing time critical applications. Apart from the comprehensive graphical elements their textual companion in Object Constraint Language (OCL) [4] can add constraints applicable to the modeling elements and additional information (i.e., initial values, derived values, certain details of operations etc.) which then helps in mapping design models to implementations and to models used for analyses. It also enables to build a bridge between the model of the application to be used for analysis and the metamodel itself.

Manuscript received April 12, 2007
T. Hussain is a PhD student at the Juniorprofessorship Agentbased Automation, University of Kaiserslautern, Erwin-Schrödinger-Str. 12, D-67653 Kaiserslautern, Germany (corresponding author e-mail: hussain@eit.uni-kl.de).
G. Frey is head of Juniorprofessorship Agentbased Automation, University of Kaiserslautern, Erwin-Schrödinger-Str. 12, D-67653 Kaiserslautern, Germany (e-mail: frey@eit.uni-kl.de).
Hence, if the model imposes some additional restrictions on the metamodel constructs it can be imposed through use of OCL constraints.

The paper is organized as follows: In section II a development process for IEC 61499 compliant distributed control applications is presented where it is also shown how and why UML diagrams are used during the development lifecycle. This section also elucidates the need for defining an UML profile for specification of IEC 61499 applications using UML and its roles. The following section then depicts the salient features of the UML profile for specifying the logical structure of an IEC 61499 application. In Section IV it is shown how UML can be used to specify execution semantics of IEC 61499 applications and how it can be used for analyses. Section V shows an application of the presented principles of modeling IEC 61499 applications using UML and how this model can be used for analyzing the applications. Finally Section VI presents conclusive remarks and an outlook.

II. DEVELOPMENT PROCESS FOR IEC 61499

To elucidate the role of UML and OCL in the development of distributed control applications compliant to IEC 61499 it is needed to present the development process to be applied. The proposed development process (cf. Fig. 1) follows the popular V process model for software development [5]. It begins with requirement elicitation by means of use case diagrams and sequence diagrams, specifying particular scenarios. The requirement specification – especially the sequence diagram – is then analyzed to get the resource entities of the IEC 61499. This step is done automatically such that a component diagram is generated where the components correspond to IEC 61499 resources.

The next step in the development process is concerned with elements of finer granularity namely, the FBs within the IEC 61499 resources. The required SIFBs for process and/or communication interfaces are already identified in the previous step and therefore the remaining constituent FBs are mainly those corresponding to the realization of control algorithms. These FBs can be user-defined as well as off-the-shelf ones. Naturally, classes are the modeling counterpart of FBs. To build a clear and unambiguous mapping, classifiers for proper class to FB mapping have to be defined. Therefore, a profile for specifying the structural elements of IEC 61499 is proposed and discussed briefly in Section III. The profile is not only the basis for building mappings between the UML and IEC 61499 constructs but also for maintaining a backward compatibility with the model for performing analysis (i.e., task models for schedulability analysis as presented in Section V).

In the second (low level) design step the behavioral models of the resources are defined as well as the behavioral model of the FBs. Considering the reactive nature of the software component UML state chart diagram is an option for specifying the behavioral model. The state diagram based specification can further be augmented by activity diagrams which give the opportunity to model control as well as object flow through the indivisible algorithmic elements in a program [6], [7].

Now as the structural and functional specification is accomplished through the use of UML it is needed to find a way of going from model to code. Here the execution semantic should play its role since it will then dictate to which run-time entity a model element can be transformed. For this purpose there needs to be a way of specifying an execution semantic. In tune with the development process and furthermore due the reasons mentioned in the previous section UML along with OCL is a good choice for this purpose. Using an execution semantics specified in such a way it is then possible to generate certain scenarios of the distributed application. These scenarios can be used for analyzing whether the application satisfies the functional and non-functional (esp., real-time) requirements. For further details of the proposed development process and its comparison with the state-of-the-art development processes for IEC 61499 compliant applications see [8].

III. STRUCTURAL SPECIFICATION

The logical structure of an IEC 61499 application can be specified in terms of certain artifacts which should strictly follow the hierarchy defined in the standard. The entities defined in IEC 61499 which are used to specify the logical
structure of an application ushers in certain concepts which have similarities as well as dissimilarities with the object oriented approach used in UML. Therefore, to specify the logical structure of an IEC 61499 compliant distributed control application the profiling mechanism of UML can be used so that the domain specific concepts like FB, Composite FB, Service Interface FB (SIFB), Resource etc. can be specified precisely. UML profiles can be defined in terms of stereotypes, tagged values and constraints. The purpose of stereotype is to show which metamodel element is typing a model element on a diagram according to the concerned profile. Stereotypes restrict the meaning of the metamodel for a certain domain. Tagged values are defined for certain stereotypes and serve as a container for property name-value pairs providing the definition of the kind of information added to the stereotype. Constraints on the other hand express certain rules of correctness applicable on any instance of the stereotypes and its tagged values. Stereotypes can be of four different types, namely, decorative, descriptive, restrictive and redefining [9]. Among these the restrictive stereotyping is chosen as it holds the semantics of the original UML metamodel. Moreover, OCL is chosen to describe additional restrictions so that these can be validated when an UML model uses a stereotype from the profile.

A list of the stereotypes and corresponding tags contained in the proposed profile is presented in Table I. Constraints applicable to the stereotypes can be of two types, namely, applicabilityConstraint that states syntactic applicability of the stereotype and actualConstraint that defines the semantic of the stereotype [10]. Fig. 2 shows the relations between the stereotype and the constraints mentioned above. The two associations between stereotype and constraint refer to the two different types of constraints since the difference is only in the role it plays and their roles are certainly exclusive as emphasized by the xor restriction. The association role that relates constraint with a modelElement refers to the role of the constraint in restricting application of a stereotype on it while the association of constraint with stereotype refers to its role of specifying precisely the meaning of the stereotype. The constraints playing the first role are applicabilityConstraint while those playing the later role are actualConstraint.

In the context of the proposed application applicability constraints are used to check the compatibility of the stereotypes with the metaclass of the elements to which the stereotype is assigned. The use of actual constraints ensures that the standard is properly followed (i.e., when data inputs are not in the list of withVars of any of the events of a FB, it must have a value for constVal tag) during the design phase. Moreover, execution semantic specific restriction can also be imposed using these constraints. For example when the sequential hypothesis of FB-execution is applied the constraints can be used to impose the restriction that when parallel branching is needed the branches cannot be in the same resource.

![Fig. 2 Relations between the constraints and the concerned stereotypes](image)

Table I: IEC 61499 Profile Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applicable Metaclasses</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;basicFB&gt;&gt;</td>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;serviceFB&gt;&gt;</td>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;compositeFB&gt;&gt;</td>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FBeventinput&gt;&gt;</td>
<td>Interface withVars</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FBeventoutput&gt;&gt;</td>
<td>Interface withVars</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FResourc&gt;&gt;</td>
<td>Component</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FBDInputData&gt;&gt;</td>
<td>Feature event, constVal, initVal</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FBDOutputData&gt;&gt;</td>
<td>Feature event, constVal, initVal</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FBDInternalData&gt;&gt;</td>
<td>Feature constVal, initVal</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;FBECCAlg&gt;&gt;</td>
<td>Operation</td>
<td></td>
</tr>
</tbody>
</table>

IV. SPECIFICATION OF EXECUTION SEMANTICS

The application domain of IEC 61499 is principally the realm of real-time distributed application and therefore it is almost always needed to ensure that certain functionalities satisfy their corresponding time constraints. In this context, the usual practice is to perform certain analyses (i.e., schedulability etc.) which need apart from the structure and behavior of the software entities the models of the logical and physical resources [11]. The physical resources consist of the processors, the networking infrastructures etc. while the concurrent processes, semaphores, queues etc. are classified as logical resources. The UML profile for Schedulability, Performance and Time (SPT) [12], [13] evolved to augment the core UML’s ability to model such resource elements and to depict the relation of the resource models to the models of the software entities.

The SPT profile is primarily divided into three packages; General Resource Modeling Framework, Analysis Models and Infrastructure Models. The first package is applicable virtually to any real-time system and is divided into three smaller sub-profiles namely, RT resource modeling, RT concurrency and RT time modeling. The first of these defines what is meant by a resource in a real-time system context, the second one defines a detailed concurrency
model and the last sub-profile defines concepts important for specifying temporal aspects. The elements of the Analysis Model package primarily deal with schedulability and performance analysis.

For the purpose of specifying execution semantics for IEC 61499 the RT concurrency modeling sub-profile is of prime importance. The central concept of the sub-profile is the ConcurrentUnit which is a kind of ActiveResource which in turn is a ResourceInstance (i.e., a run-time entity whose services can be characterized by QoS values) defined in the RT resource modeling framework. The notion of ConcurrentUnit closely corresponds to the notion of thread and it has a StimuliQueue which in the context of IEC 61499 applications can be specified to be an EventQueue. An overview of the principal concepts of the concurrency profile applicable to IEC 61499 is shown in Fig. 3.

The components shown in Fig. 3 constitute the metamodel for mapping from the domain of IEC 61499 compliant distributed control applications to corresponding resources needed from operating systems (OS) to execute an application. This metamodel is helpful in two ways: Firstly, to convert from IEC 61499 elements to the platform dependent codes and secondly, to perform certain analyses to check whether a conceived distribution of application components can provide satisfactory performance especially in dealing with hard real-time constraints. For the first purpose further hints for implementing artifacts like ConcurrentUnit, EventQueue are needed in the parlance of the programming languages and this is not in the scope of the current paper. For the second purpose, i.e. to perform the analyses, it is needed to acquire modeling artifacts as well as certain relevant information like worst case execution time of the operations of the artifact.

The OCL statements state that each of the SIFBs and timer FBs should be allotted ConcurrentUnits while the remaining FBs are passive objects which receive synchronous calls and are executed by the ConcurrentUnit that makes the call. These calls are actually the internal events generated by the sequence of EC actions performed due to the external triggers. This later mapping is then explicated using another set of OCL transformations.

V. EXAMPLE

To apply the concepts as well as to observe their merits and demerits, the development of a distributed control application for a didactic plant is considered. The didactic plant is divided into four modules as shown in the schematic diagram of Fig. 4. During the normal mode of operation cylindrical objects are checked for appropriate thickness, and then a hole is drilled in them which is later tested for propriety. Finally, the objects are sorted into repositories depending on their types.
An IEC 61499 compliant distributed control application should be developed for this didactic plant. Especially, the development of control application for the third and the fourth module of the plant is considered in the current context. Sequence diagrams depicting scenarios for the normal mode of operation are depicted in Fig. 5. The sequence diagrams show how the rotary motor controller and gantry crane motor controller should react in response to sensor signals. These two scenarios can run in parallel such that a processed object delivered from module 3 can be transported by the crane of module 4 while the rotary table rotates due to appearance of a new object at position 1. It deserves mention that in both these scenarios there are strict deadlines which have to be met. In the rotary motor control scenario the rotary motor controller has to stop the motor and the time span between the instant when the sensor detected the presence of the rotary table above a position and the instant when the motor is stopped must not be more than 3 ms. Similarly, in case of the second scenario the time elapsed between the instant when the arrival of the crane upon a position is sensed and the instant when the motor is stopped must not be more than 10 ms.

While developing the controller an obvious choice could be to assign an IEC 61499 resource to encapsulate the control functionalities of each of the modules. Therefore, the FBs implementing the control functionalities as well as the corresponding service interfaces for module 3 can be encapsulated in one resource and similar organization of FBs will be applied to the FBs of module 4. The SPT annotated UML sequence diagram of Fig. 6 depicts the interactions of SIFBs and FBs required for control of the rotary motor. If a different situation is considered where a slightly different arrangement of the FBs is needed to be compared to the above mentioned arrangement of the FBs, it will be needed to perform schedulability analysis to check out whether the time constraints will be met in the new scenario or even to check out a quantitative comparison of the scenarios. For example, it could possibly be conceived that the control functionalities of rotary motor and crane be arranged under the same IEC 61499 resource (cf. Fig. 7). This has certain advantages, such as, only one instance of timer will suffice for this purpose and if the sensor signals are physically read from the same memory block of a controller module a single SIFB can be used to read the sensor values. This can bring advantages from the temporal perspective, especially, since both the control functionalities are to be accomplished fulfilling strict deadlines.

Using the annotated UML models schedulability analysis of IEC 61499 resources on a single physical device has been carried out. Since the resources are assumed to be NPMT, when two resources are deployed on one device (cf. Fig. 7 (top)) the tasks of the two resources need to be switched to maintain concurrency. It is assumed that this will be done after completion of each of the critical region executions of the resources. NPMT implementation also implies that parallel branches will be executed branch by branch. Additionally, it is assumed that the branch with more strict time constraints will be prioritized to execute before the other parallel branches. The results have shown
that allocating FBs related to the control of a particular module to the same resource fails to meet the shortest deadline namely that related to the response of the motor controller of the rotary table. The allocation of motor and crane control module to the same resource and all the other non-critical control FBs into another resource (cf. Fig. 7 (bottom)) results in satisfaction of both the deadlines. This is mainly due to the use of the same thread for periodic triggering and reading from the memory block where sensor values are written down at regular intervals. Moreover, the interception of the motor controller’s operation by the control program residing in the other resource of the same device and the resulting context switches causes delays that sum up to more than 3ms.

VI. CONCLUSIONS AND OUTLOOK

To reduce the diversity of interpretations and to avoid the resulting misconception or to resolve certain compliance issues among the developers of IEC 61499 applications, there need to be an execution semantic defined. The ongoing works of deciding on a semantic profile should certainly lead to a resulting execution semantic which will not be a very restrictive one but a set of semantic profiles defining the guidelines for building the executive deliverables from the modeling elements. In this context it could be often needed to perform certain analyses on alternative implementations to check whether time constraints are satisfied. The proposed modeling concept of the platform specific models should be a useful tool from this perspective.

For complex applications it is often not possible to decide early in the development which particular execution semantics will show optimal performance or is able to satisfy all the time constraints in the first place. In these cases, the proposed modeling concept will provide a flexible link between modeling and analysis.

REFERENCES