U.P.B. Sci. Bull., Series C, Vol. 85, Iss. 4, 2023 ISSN 2286-3540

# ADDRESSING CHALLENGES IN MODELING REAL-TIME APPLICATIONS FOR COMPLEX DISTRIBUTED SYSTEMS

Liliana DOBRICA[1]

*The paper explores the current challenges in software modeling of real-time applications for complex distributed systems taking benefits of the current abstracted mechanisms and graphical elements supported by UML and its MARTE profile. The main concerns are focused on modeling the structure and behavior of a set of various types of tasks that are executed concurrently, modeling asynchronous events including a more detailed representation of the hardware interrupts mechanism, modeling timing constraints in structural diagrams, scheduling policies, concurrency on shared resources and intertask communication mechanisms. A model-driven development approach based on open-source tools integrated in Eclipse platform is used to address the modeling challenges.*

**Keywords**: real-time software design, model-driven development, UML, MARTE, modeling tools, open-source.

## 1. Introduction

As new software applications have emerged over the last decade, the real-time notion has evolved for the complex distributed systems (CDS) represented in the various forms including Internet of Things (IoT) or Cyber-Physical Systems (CPS). Essentially, the time is not only an issue of an application performance, but a critical concern for a correct functional execution that should be engineered properly in a coherent and correct manner. Many of the system development cycle challenges are addressed during the design and analysis stages using appropriate modeling technologies. Appropriate modelling approaches need to bridge solutions of the key problems of the general-purpose computing with the concurrency, heterogeneity, and sensitivity to timing of specific real-time requirements for such applications.

Real-time software modeling for a CDS having functionalities to monitor and control specific elements of the operation environment is not a trivial activity [1]. Among the key challenges is the selection of proper methodologies and software tools for modeling the real-time software of complex, coordinated and time sensitive distributed systems that meet the demands of the certification standards [2]. When a complex behavior is characterized by concurrency control, synchronization mechanisms, distributed processing, and nondeterministic

[1] Professor, Dept. of Automation and Applied Informatics, UNST POLITEHNICA of Bucharest, Romania, email: liliana.dobrica@upb.ro

timings, the use of formal modeling languages increases the confidence in the quality of the expected results and the degree of rigor since early stages of the development cycle.

Recently many research efforts have been focused on integrating Unified Modeling Language (UML) specifications in the software development cycle in industrial context [1]. UML has been defined with the ability to be customized through its profiling mechanisms to directly support specific concepts of an individual application domain including real-time constraints [3,4]. Nowadays, the industry uses UML models for understanding, refinement and improvement of the system design models. A plethora of methodologies provide guidelines to specification, modeling and design of real-time CDS [5,6]. These propose different and complementary architectural viewpoints to be considered in the system functionality and quality definition at architectural level and allow a proper management of the system complexity. Decomposition into smaller elements and their refinement reduces and manages complexity. MARTE is the UML profile for modeling and analysis of real-time and embedded systems that provides a wide support for specification, design, verification and validation of complex systems. [7,8,9]. The modeling concepts provided by MARTE profile can be used in the software development life-cycle from requirements specification to detailed design, contributing in this way to a better communication between stakeholders, standardization and interoperability between platform tools developers. The presence of MARTE profile annotations in the architectural design model can contribute considerably to the correctness, effectiveness and suitability of the model towards addressing stakeholders needs and expectations.

The purpose of this paper is to address several challenges of modeling real-time applications for CDS taking the benefits of the current abstracted mechanisms and graphical elements supported by UML and its MARTE profile. The exploration starts with a discussion about the current model-based and model-driven software development approaches, then it continues with the presentation of challenges in using open-source modeling tools. The next sections present the main concerns of the real-time applications modeling for the current CDS and provide specific solutions addressing the key challenges. These research objectives are to provide models of the structure and behavior of a set of various types of tasks that are executed concurrently, identify asynchronous events including a more detailed representation of the hardware interrupts mechanism, describe how to formalize the timing constraints in the structural diagrams, scheduling policies, concurrency on shared resources and inter-task communication mechanisms. A model-driven development approach based on open-source tools integrated in Eclipse platform is considered to address the modelling objectives.

## 2. Background

### 2.1. Development approaches

Model-based systems engineering (MBSE) is an approach that implements a part or the whole engineering process of a system starting from requirements by using appropriate modeling languages. The product is an integrated model that consists of a structural model, a behavior model of the functionalities, a performance model and other analysis models.

When MBSE is compared with the traditional V-model of the software development lifecycle [2], among the main benefits to be mentioned are a better communication, reduced development risks, quality improvements, increase of productivity and better knowledge transfer. Communication offers a better understanding among the development teams by providing interoperability and the ability to integrate many viewpoints representing various stakeholders' concerns. Continuous validation and verification of the requirements and a precise estimation of the development costs reduce the development risks. Better quality is obtained from having completeness and no ambiguity of the requirements that can be easily verified with test cases and can be traced during the following development stages including design and analysis. Increase of productivity is based on shorter analysis of the requirements changes, reusing of existent models, reducing errors and integration time. A standard format of the design description can be easily accessed improving the knowledge transfer.

The modeling effort depends on the purpose which varies depending on the life stage of the system that can be under development or in operation. When a new system is created the modeling effort is on the specification, analysis and design. An existent system can be modeled aiming to provide a characterization of its components and behavior for training its users about operation or maintenance. The modification of an existent system involves a modeling effort, too. The breadth, depth and fidelity of a model are considered to define or match the scope of the model, its completeness or its understanding. A model has five key characteristics: abstraction, understandability, accuracy, predictiveness and inexpensiveness [5]. Moreover, a model should conform to the rules of a modeling language to ensure its consistency. The modeling tool should enforce these rules and provide a report of violations.

Model-driven engineering (MDE) is a software engineering community initiative suggesting that one should first develop a model of the system under study, which is then transformed into an executable software entity. It is a field of MBSE in which the process is relying on the production and use of models in a disciplined and rationalized manner. The system model can target different execution platforms with different operating systems and hardware resources. It includes model-to-model transformations and model-to-text-transformations, platform independent models and platform specific models [10,11,12]. The abstraction, interoperability and reusability capabilities of MDE are very relevant to model complex services built on distributed, heterogenous devices.

Current modeling methodologies are continuously improved. From a model it is possible to analyze, explore different solutions and optimize a system towards satisfying all its requirements [13,8]. Executable UML is the current research trend that started with the introduction of UML2.0. A formal specification with the executable semantics for a subset of UML2, via the Foundational Subset For Executable UML Models (fUML) have been defined [14]. A compact and complete specification of complex behaviors including algorithmic parts combines fUML and a textual action language ALF (Action Language for Foundational UML) [15] such that the models based on these can be validated and executed.

### 2.2. Modeling UML projects with open-source tools

The use of an open-source tool support for the design and analysis of software for CDS is currently a preferred practice in industry or academic context, due to the many benefits provided by this approach [16]. One of the most important benefit is the gain of independence from the proprietary tool vendors, that can change the conditions under which a tool is commercialized and maintained and many times these changes can impact a lot on the customers. Companies or universities involved in software systems modeling, for teaching or research activities, have experienced many challenges. Such challenges include vendor lock-in, lack of tools maintenance, tool and service acquisition by another company or changes in the academic program of the tool vendor. Among solutions to overrun these challenges is to contribute to research and development of open-source software (OSS) projects that are designed to be extensible. Universities' teams with interest in research and technology join this partnership in an ecosystem and work together towards the development of a base, common OSS platform [17,16]. Companies can later build on this OSS platform to develop their own customized solutions or to provide added value features and services. Many large companies are supporting this vision of OSS tools for systems engineering, too.

Eclipse Papyrus is an OSS tool that provides an environment to be used in MBSE or MDE approaches. Similar to many other OSS modeling tools for systems or software development, it is based on open standards as a reference for modeling notation. Eclipse Papyrus has been used in many industrial projects including design and evaluation of CPS or other CDS with real time constraints. For several years this tool has become a valid choice in our academic context for both teaching students and research purposes. Recently, it has been released the version 6.3.0. of this modeling environment [18]. Papyrus integrates technologies for a graphical editor that implements all the diagrams of the UML 2.5 standard specification as defined by OMG [4]. This modeling environment provides also support with specific tabular and graphical editors required by SysML notation to enable MBSE approach. The latest version, SysML 1.6, is available on the Eclipse Market Place [17].

Papyrus Moka for Eclipse is another modeling tool that can be integrated in this environment. It is used to provide execution, debugging and animation of a modeled system [18]. These three capabilities distinguish by the following characteristics:

- In execution, when an UML model behavior is represented by composite structure, activity or state machines diagrams can be executed to assess if the system will behave as expected.
- In debugging, when the modeled system does not behave well, the Moka debug framework enables the user to control the execution of the model. It can suspend the current model execution and then the user can observe if the associated values of defined variables are appropriate.
- In animation, the execution flow of an application model can be visually followed by the user of the tool by activating the animation capability. Thus, the user can see how the model elements on diagrams can change their visual representation when executed.

There exist other technologies that use Papyrus, extend it or complement it for specialized aspects. Among these it can be mentioned Papyrus-RT for Real-time Systems Modeling, Papyrus for Robotics or other Eclipse UML profiles in Repository. Eclipse UML profiles Repository includes technologies under an Eclipse MDT sub-project, most of them in the incubation phase, where various standards including standardized UML profiles have been developed. Here there is no centralized repository and sometimes there are incompatibilities between different tools. The implementation of the MARTE profile is in this category in the incubation phase.

### 2.3. Real time applications modeling for the current CDS

The main concerns in real-time applications modeling are related to scheduling of tasks and latency requirements. A real-time CDS has to support arbitrary release times for their tasks and deadlines, scheduling under various resource constraints, preemption or minimization of jitter, predictability of the consequences of every scheduling decision, reliability in the face of failures and partitions, robustness in the face of overload conditions, tasks with different levels of criticality, approximation of computation to meet hard deadlines. Many real-time CDS have low latency requirements because a small increase in system latency may significantly cost the business using the CDS [19,20].

The main challenges in modeling real-time CDS include the realization in a consistent manner of the entire model and the management of concurrency in structural modeling, behavior modeling, state-machine models or component diagrams. Quality assurance is represented by the analysis of the CDS model based on specific methods and tools [10,2]. The purpose of the structural modeling is dependent on the decomposition hierarchy describing the problem scope, interfaces between hardware and software components and defining the software system scope and components. Behavior modeling considers the interaction sequences among software model elements. State-machine models provide a perspective on the reactions to external input events of the model elements. Concurrency management refers to modeling activities that are performed in parallel in a specific manner to manage multiple input sequences or unpredictable processing loads. A distributed processing environment needs to model a deployment view describing how is the allocation of the software components to various processing nodes. Model analysis before stepping into the next development stage indicates whether the entity of interest will be able to fulfill the intended purpose or is suitable towards addressing the stakeholders needs and expectations [21].

Working with a solution defined by a platform-based modeling concept gives good results. This is represented in Fig. 1 and a detailed description is provided in [7] where the platform is "the full complement of hardware and software that underlies and supports an application".

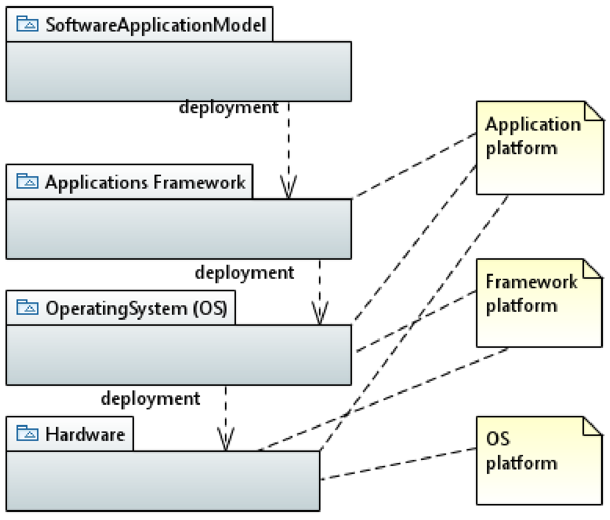

Fig 1. The platform concept of MARTE [17]

## 3. Addressing challenges in modeling a real-time CDS

Problems that can arise in modeling a real-time application for CDS are defined by the concurrency, heterogeneity and timing constraints. The real-time CDS model consists of a set of concurrent and interacting tasks that can be described graphically in a UML class diagram. The model of a task is to release jobs in a recurrent manner with periodic, sporadic or jittering policies depending on whether the release time is deterministic, bounded by a minimum but not maximum value, or constrained between a minimum and a maximum value, respectively. A job is a sequence of computational blocks, each characterized by a nondeterministic execution time constrained between a minimum and a maximum value. Computational blocks are associated with an entry-point for the attachment of a function method to the corresponding low-level implemented component. Computational blocks belonging to jobs of different tasks may have semaphore synchronization and message passing communication. Also they may require computing resources, in which case they are associated with a priority level and run under static priority preemptive scheduling.

### 3.1. Addressing challenges in structural modeling

Structural modeling of CDS with timing constraints uses UML class diagrams with MARTE profile. From an application designer perspective, the main concerns are to identify the elements that represent concurrent tasks and to specify application meaningful information concerning their concurrency properties such as priority and deadline. MARTE stereotype used to represent concurrent tasks is «SWSchedulableResource» that is applied to UML active classes.

In Fig. 2 *Task* is an active class and a «SwSchedulableResource». The *type* of the «SwSchedulableResource» is *ArrivalPattern*, that is provided by MARTE being used to specify real-time characteristics of a task for analysis. Also, the *Task* class is a shared superclass taking the benefit of the UML generalization-specialization mechanism to avoid duplication of definitions, while still maintaining separation between three alternatives. The CDS is modeled in an event-driven manner having both external events arriving to the system and internal events generated by each task at its competition. The task system has three types of tasks based on the different kinds of external events arriving at the system: periodic, sporadic and singular. A *PeriodicTask* has an execution period and is activated by a stream of events that are generated periodically. A *SingularTask* is activated by an event that is generated only once. A *SporadicTask* is activated by a stream of events that have a minimum and maximum interarrival times, which are the minimum time and, respectively, maximum time between the generation of two events. Event classes are represented in MARTE by the different arrival patterns as described in Fig. 2 in the *ArrivalPatte*rn.

Many important properties of the application model element are provided by the «SwSchedulableResource». The list of properties includes: *entryPoints*, *priorityElements*, *stackSizeElements*, *heapSizeElements*, *activateServices*, *resumeServices*, *suspendServices*, *terminateServices*, *deadlineElements*. Some of these are specified in the application model being relevant for the required service to the platform that offers a quality level of its service.

A platform view, particularly of the operating system designer, has relevant concerns for a behavior including *activateService*, *suspendService*, *resumeService* and *terminateService* that are defined in the state machine model of a task.

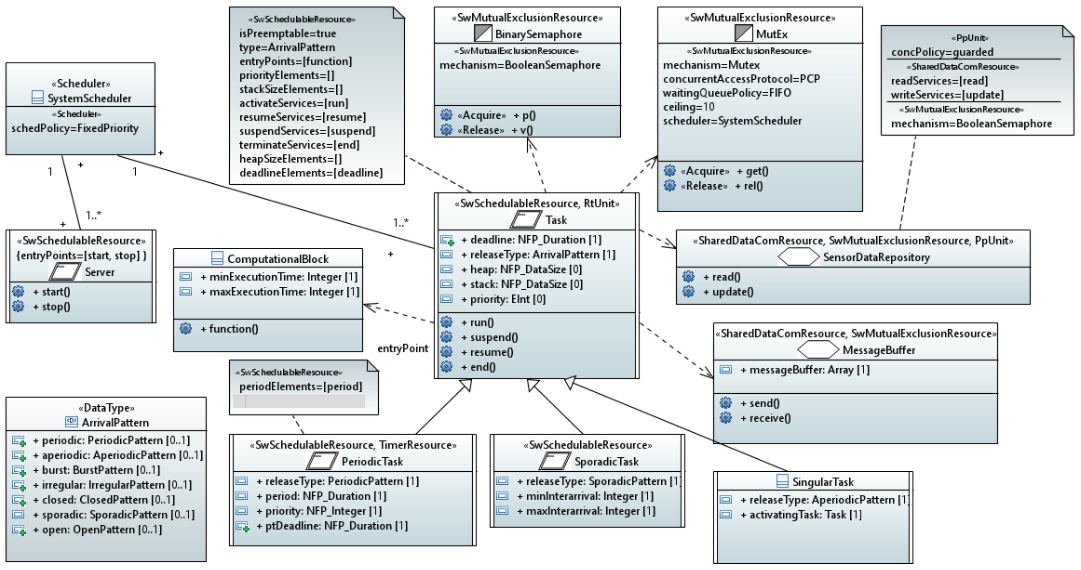
«SwSchedulableResource»
isPreemptable=true
type=ArrivalPattern
entryPoints=[function]
priorityElements=[]
stackSizeElements=[]
activateServices=[run]
resumeServices=[resume]
suspendServices=[suspend]
terminateServices=[end]
heapSizeElements=[]
deadlineElements=[deadline]
«SwMutualExclusionResource»
BinarySemaphore
«SwMutualExclusionResource»
mechanism=BooleanSemaphore
«Acquire» + p()
«Release» + v()
«SwMutualExclusionResource»
MutEx
«SwMutualExclusionResource»
mechanism=Mutex
concurrentAccessProtocol=PCP
waitingQueuePolicy=FIFO
ceiling=10
scheduler=SystemScheduler
«Acquire» + get()
«Release» + rel()
«PpUnit»
concPolicy=guarded
«SharedDataComResource»
readServices=[read]
writeServices=[update]
«SwMutualExclusionResource»
mechanism=BooleanSemaphore
«Scheduler»
SystemScheduler
«Scheduler»
schedPolicy=FixedPriority
1
+
1
+
1..*
+
1..*
«SwSchedulableResource»
{entryPoints=[start, stop] }
Server
+ start()
+ stop()
ComputationalBlock
+ minExecutionTime: Integer [1]
+ maxExecutionTime: Integer [1]
+ function()
+
entryPoint
«SwSchedulableResource, RtUnit»
Task
+ deadline: NFP_Duration [1]
+ releaseType: ArrivalPattern [1]
+ heap: NFP_DataSize [0]
+ stack: NFP_DataSize [0]
+ priority: EInt [0]
+ run()
+ suspend()
+ resume()
+ end()
«SharedDataComResource, SwMutualExclusionResource, PpUnit»
SensorDataRepository
+ read()
+ update()
«SharedDataComResource, SwMutualExclusionResource»
MessageBuffer
+ messageBuffer: Array [1]
+ send()
+ receive()
«SwSchedulableResource»
periodElements=[period]
«DataType»
ArrivalPattern
+ periodic: PeriodicPattern [0..1]
+ aperiodic: AperiodicPattern [0..1]
+ burst: BurstPattern [0..1]
+ irregular: IrregularPattern [0..1]
+ closed: ClosedPattern [0..1]
+ sporadic: SporadicPattern [0..1]
+ open: OpenPattern [0..1]
«SwSchedulableResource, TimerResource»
PeriodicTask
+ releaseType: PeriodicPattern [1]
+ period: NFP_Duration [1]
+ priority: NFP_Integer [1]
+ ptDeadline: NFP_Duration [1]
«SwSchedulableResource»
SporadicTask
+ releaseType: SporadicPattern [1]
+ minInterarrival: Integer [1]
+ maxInterarrival: Integer [1]
SingularTask
+ releaseType: AperiodicPattern [1]
+ activatingTask: Task [1]

Fig. 2. Structural view model of the application

Modeling an application includes design decisions about scheduling policies, synchronization mechanisms and the resource-based model of mutual exclusion. Scheduling policies are established by the *SystemScheduler* that is a MARTE «Scheduler» stereotype. The attribute *schedPolicy* is for *FixedPriority* scheduling algorithm This is a design choice from a list of *SchedPolicyKind* which is an enumeration of the following: *EarlistDeadlineFirst*, *FIFO*, *FixedPriority*, *LeastLaxityFirst*, *RoundRobin*, *TimeTableDriven*, etc. For instance, using fixed priority algorithms different scheduling strategies are allowed, including preempted and non-preempted scheduling or interrupt service routines, sporadic server scheduling or periodic polling servers.

Concurrency conflicts to shared resources are specified with «SwMutualExclusionResource» stereotype that support mutual exclusion techniques. The application model includes a *BinarySemaphore* and *MutEx*. A *mutex* is similar to a binary semaphore, but the difference is that it can only be released by the task that currently holds it. The concurrent access protocol for the mutex is *PCP*, that is a priority ceiling protocol. By selecting this protocol, the ceiling priority value has to be specified in the ceiling attribute. PCP regulates the access to shared resources by raising the priority of any locking task to the highest priority of any task that ever uses that lock. The *BinarySemaphore* class has the *p()* and *v()* operations to access and, respectively, to release the semaphore. The model uses stereotypes «Acquire» and «Release».

The complex mutex that uses a ceiling protocol and handles the incoming acquire requests based on FIFO waiting policy is represented in the application model. The *binarySemaphore* is used to access *SensorDataRepository*. Additionally, the concept *protected passive unit* («PpUnit») is used in the application model represented by the stereotype. The *concPolicy* attribute is a guarded value because it involves mutual exclusion such that only a single concurrent access is allowed while others are blocked until their turn comes. Two basic communication techniques between tasks, shared data communication and message-based communications are abstracted in MARTE and can be used in modeling applications (see Fig. 2).

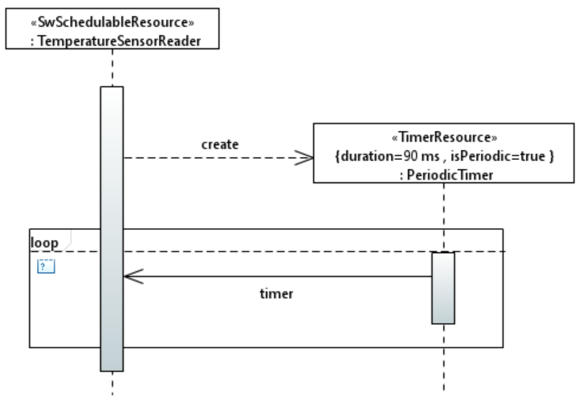

Fig 3. PeriodicTimer creation and interaction

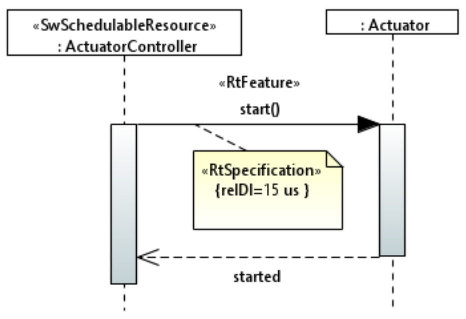

Fig. 4. Timing information in the sequence diagram

Shared data repositories support the exchange of information between tasks and «SharedDataComResource» stereotype represent these. Message-based communication involves placing information generated by a source task into a message buffer which is then delivered to the destination task. Conceptually the last technique involves the movement of data from one location to another through the network of a distributed system. In the application design it could be used for interaction synchronous operation calls or asynchronous signals.

### 3.2. Addressing challenges with timing constraints and asynchronous events

The application modeling with MARTE includes modeling timing mechanisms such as clocks and timers for representation of timing constraints. A clock interrupt may trigger a computational block and a periodic timer may be used to awake periodic tasks. Also, time is associated with behavior by specifying timing information that represents either timing requirements, such as deadlines, or timing properties such as execution durations.

Fig. 3 illustrates the occurrence of a periodic timeout event by an accurate explicit timing mechanism with the TemperatureSensorReader task that creates a PeriodicTimer object that sends back to its creator an asynchronous timer message every 90 miliseconds. The «TimerResource» stereotype indicates a cyclical behavior with specified periods.

Timing information needs to be specified when invoking services in a behavior view described with a sequence diagram or activity diagram. In the sequence diagram «RtFeature» and «RtSpecification» stereotypes are used together for an operation call as illustrated in Fig. 4 for the start operation that should be completed within 15 microsecs. It specifies relDI, of 15 microsecs, which is the relative deadline starting from the instant the invocation is initiated.

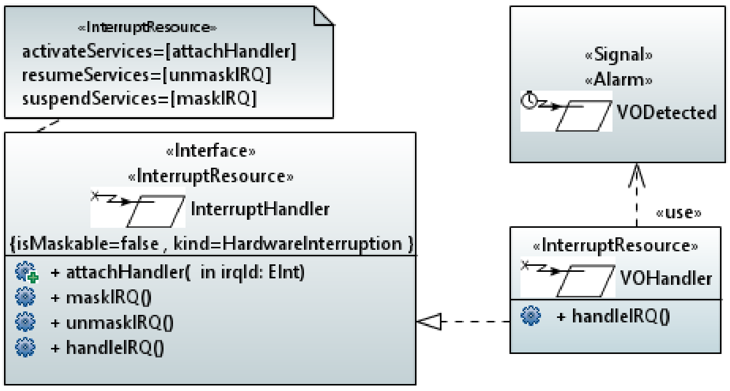

Fig. 5. Capturing common characteristics of an interrupt pattern

Fig. 5 describes the use of «InterruptResource» and «InterruptHandler» interface stereotype that captures common characteristics of interrupt system and handlers for an interrupt signal generated by detecting an anomalous situation in the monitored environment, (VO - Voltage overload). High urgency events are handled in a timely fashion based on the interrupt pattern. These events represented by hardware signals are handled even when the system is busy doing other processing.

The watchdog timer is modeled with two stereotypes, «Alarm» and «TimerResource» as illustrated in Fig. 6. The Alarm stereotype is a source of a hardware interrupt, and it could be used to detect and prevent livelocks and deadlocks in real-time operating systems [17] by generating and dispatching signals asynchronously to targeted concurrent tasks.

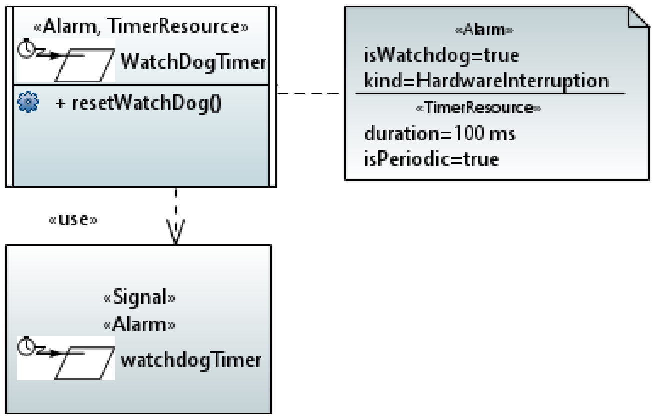

Fig 6. Modeling a WatchDogTimer

## 4. Conclusions

The paper has described the main results of the current research in open-source modeling technologies to be used when considering the evolution of the real-time constraints of the software applications for existing CDS. When heterogeneity, concurrency and timing constraints are the key properties of the existing CDS, modeling software applications for such systems poses many challenges that have been explored. Specific modeling solutions addressing the key challenges have been advanced including models of the structure and the behavior of a set of various types of tasks that are executed concurrently. UML with MARTE profile is a promising solution of a formal representation for timing constraints in the structural and behavior diagrams, scheduling policies, concurrency on shared resources and inter-task communication mechanisms The open-source Eclipse Papyrus tool is very useful and a suitable tool to realize an application software model project in UML. The tool provides a modeling perspective that can be configured to explore the UML model elements. Valuable results have been obtained in specifying the application model elements and applying MARTE profile after importing it. The tool has been used to generate code in JAVA after creating the specific model by annotating model elements with JAVA stereotypes. The model can be validated and executed when the behavior of an active class is specified by an activity diagram.

## R E F E R E N C E S

[1] *I. Bicchierai, G. Bucci, L. Carnevali and E. Vicario*, "Combining UML-MARTE and Preemptive Time Petri Nets: An Industrial Case Study", IEEE Trans on Ind. Inf, **9**(4), 2013.

[2] *L.Carnevali, L. Ridi and E. Vicario*, "Putting Preemptive Time Petri Nets to Work in a V-model SW Life Cycle", IEEE Transactions on Software Engineering, **vol. 37**, no. 6, 2011.

[3] *D. Du, P. Huang, K. Jiang and F. Mallet*, "pCSSL: A stochastic extension to MARTE/CCSL for modeling uncertainty in Cyber Physical Systems", Science of Computer Programming, **vol. 166**, pp. 71-88, 2018.

[4] *** UML, http://www.uml.org (accessed on May 2023)

[5] *G. Harbour, G. Garcia, P. Gutierrez and D. Moyrano*, "MAST: Modeling and Analysis Suite for Real Time Applications", Euromicro Conference on Real-time Systems, pg. 125- 134, 2001.

[6] *E. Villar, J. Merino, H. Posadas, R. Henia and L. Rioux*, "Mega Modeling of Complex Distributed, Heterogenous CPS Systems", Microprocessors and Microsystems, **vol. 78** (2020).

[7] *B. Selic and S. Gerard*, Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE, Ed. Elsevier, 2014

[8] *F. Cicozzi, I. Malavolta and B. Selic*, "Execution of UML models: a systematic review of research and practice", Software and Systems Modeling, **vol. 18**, 2313-2360, 2018

[9] *F.G. Ribeiro, A. Retberg, C. Pereira, C. Steimetz and M. Soares*, SPES Methodology and MARTE Constraints in Architectural Design, IEEE Symp. on Comp and Comm., 2018

[10] *L. Dobrica*, "Exploring Approaches of Integration Software Architecture Modeling with Quality Analysis Models", 9th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2011.

[11] *R. Mzid and M. Abid*, "UML-Based Reconfigurable Middleware for Design-Level Timing Verification in Model-Based Approach", 11th International Design and Test Symposium, pp. 181-186, 2016.

[12] *A.Bennett and A.J. Field*, "Performance Engineering with the UML Profile for Schedulability, Performance and Time: a Case Study", Procs of MASCOTS, 2004.

[13] *L. Dobrica*, "Quality of Transformations providing Interoperability in Software Architecture Model-Driven Development", 6th International Conference on Software and Database Technologies (ICSOFT), pp. 305-308, 2011

[14] *** fUML, http://www.omg.org/spec/fuml (accessed on May 2023)

[15] *** alf, http://www.omg.org/spec/ALF (accessed on May 2023)

[16] *J.S. Vara, A. Ruiz and G. Blondelle*, "Assurance and certification of physical systems: The AMASS open source ecosystem", Journal of Systems and Software, **vol. 171** (2021).

[17] *** Eclipse, https://www.eclipse.org/projects/ (accessed on May 2023)

[18] *** Papyrus, https://www.eclipse.org/papyrus/ (accessed on May 2023)

[19] *A. Kejariwal and F. Orsini*, "On the definition of real-time", 2016 IEEE TrustCom-BigDataSE-ISPA, pp. 2213- 2220, 2016

[20] *E. Ovaska, L. Dobrica, A. Purhonen and M. Jaakola*, "Exploration of Tehnologies for Autonomic Dependable Service Platforms", 6th Int. Conf on Software and Database Technologies (ICSOFT), pp. 115-124, 2011

[21] *I. Traore, I. Woungang, A. Ahmed and M. Obaidat*, "UML-Based Performance Modeling of Distributed Software Systems", Procs of 2010 Int. Symp. of Performance Evaluation of Computer and Telecommunication Systems, pp. 119-126, 2010.