We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Development of Multi-Agent Control Systems using UML/SysML

Iskra Antonova and Idilia Batchkova
University of Chemical Technology and Metallurgy, Department of Industrial Automation
Bulgaria

1. Introduction

The successfully implementation of control systems depends greatly on achieving lower development costs, short time of design and high level of safety and reliability. However the growing amount of ICT applications in the field of automation and control, from another side, increases rapidly the size and complexity of the software systems used in this domain. The main trends in automation are connected with the development and implementation of reconfigurable distributed control systems consisting of multiplicity non-hierarchical modules linked together via different types of communication systems. In order to control the complexity of distributed real-time systems the following main requirements have to be fulfilled: reliable concept for decomposition and modularity, openness for extensions in the cases of new products, machines and devices, the general architecture model of the system has to be producer independent, use of encapsulated, reusable components.

To successfully meet the new requirements to the developed distributed control systems various modelling approaches and methods are increasingly resorting to be used. The control functions in distributed systems can be suitably described with any discrete-event model like automata, Petri nets, Grafcet/SFC, formal languages or IEC-61499 based Function blocks. The major drawback of these approaches is the lack of inheritance and polymorphism that may limit the reusability in non-vital applications. In order to overcome these shortcomings new approaches for software system development in control and automation are needed. Many working groups are trying to fill the gap between state of the art in software engineering and state of the practice in the control application domain through the use of object-oriented and agent-oriented modelling techniques.

The agent community has considerable interest in developing methods and techniques for specifying, modeling, implementing and verifying of Multi-Agent Systems (MAS) for distributed control, but so far no standardized design methodology has been recognized. The successfully mission of Unified Modelling Language (UML) [OMG-UML, 2010] as unification of numerous different object-oriented approaches led to the idea of applying UML to the design of MAS. From a software point of view, agent-based systems can be regarded as a specialization of object-oriented systems but UML does not provide enough means for capturing all agents related modeling aspects like autonomy, pro-activity and cooperation. The software agents have their own thread of control, localizing not only code and state but their invocations as well [Parunak, 1999]. In an open and distributed, agent-based, integrated control environment, the need of standard mechanisms and specifications are vital for ensuring the interoperability of the autonomous agents.

The approach presented in this chapter aims to extend UML for development of multi-agent control systems based on its profile for System Engineering – SysML [OMG-UML, 2003], [OMG SysML, 2006], the upcoming IEC-61499 standard defining the basic concepts and reference architecture and models for design of modular, re-usable, open and vendor independent distributed control applications [IEC-61499, 2005], and the FIPA's reference model for agent communications [FIPA, 2010]. SysML is the first UML profile for system engineering, which supports the specification, analysis, design, verification and validation of a broad range of complex systems. Using this profile, essential aspects of control engineering issues such as concurrency, hardware architecture, and requirements traceability can be applied to multi-agent and embedded systems. The customization of UML for systems engineering supports modeling of the whole closed loop control system including hardware, software, data, procedures and facilities. There is a clear correspondence between UML/SysML and IEC-61499 standard. The modelling concepts in both standards share many similarities and this allows the mapping between basic elements from IEC-61499 and UML/SysML profile using additional stereotypes. In order to extend the standardized concept to the whole development life cycle of a control application and to use the all benefits of object-oriented paradigm, the concepts of IEC-61499 standard are extended with different UML/SysML notations. The UML/SysML profile is extended and at the same time restricted with different stereotypes from FIPA communication standard in order to fill the gap from absence of unified communication protocol in applications based on the IEC-61499. The basic idea is to create an executable functional model of the software at an early stage of the development process based on requirements specifications, possibility for formal specification, verification and validation of the system based on modified Harmony SE methodology.

The chapter is organized in 6 parts. After this introduction, in part 2 the current trends in automation and control domain are discussed. In part 3 the main features of multi-agent systems in respect to the development of multi-agent control systems are summarized and the approaches, methods and tools for their achievement are shortly described. Part 4 of the chapter analyses different extensions of UML for modelling of hard real time systems including the special topic of agent-based control systems. As the suggested approach is based on the collaborative use of UML and the reference architecture and models proposed in the IEC-61499 standard, a short overview of these models is presented in this part too. In part 5 the proposed approach for development of multi agent control systems based on the UML profile for system engineering - SysML, and the standardized concepts, models and architectures of IEC-61499 and FIPA standards is described. A modified methodology based on a subset of the large methodology for integrated system and software development process Harmony-SE is used. The development process includes the following basic cycles Requirements Analysis, System functional analysis, Architecture design and Hardware/Software design specification supported by different UML/SysML notations and diagrams, restricted according IEC-61499 and FIPA standards. A case study, illustrating the suggested approach is presented in part 6. This example shows the control system development processes to the FESTO Distribution station. Finally some conclusions are made.

2. Current trends in the automation and control domain

The global competition between enterprises and the new production strategies adopted by business require new types of automation and control systems, characterized by a high degree of horizontal and vertical integration and self-organizing features to enable rapid

adaptation to changes in the environment. To meet the challenges of modern enterprises, automation and control systems must meet certain requirements, which can be summarized as follows [Bussmann&McFarlane, 1999]:

- The architecture of the control should be decentralized and product- / resource- based;
- Control interactions should be abstract, generalised and flexible;
- Control must be proactive and reactive;
- Control must be self-organizing.

The main challenges facing the field of automation and control are associated with a high degree of flexibility and adaptability to the emergence of unexpected needs and the wide variety of products with a view to achieving the agility of manufacturing systems. These requirements can be met by control systems that have reduced complexity, increased flexibility and adaptability in real time, are extendable, heterogeneous and support autonomous operations. The architecture of control systems has to ensure production agility through rapid adaptation and cheap changes in production environment.

To fulfil the requirements for agility, control systems of new generation are challenged to solve two major and diametrically opposed tasks:

- Full integration of information and control systems in order to automate the entire product lifecycle;
- Maintenance of the relative autonomy of the elements of the production system through transition from centralized to decentralized, distributed structures of control and information processing.

The main trends in automation are connected with the development and implementation of reconfigurable distributed control systems consisting of multiplicity non-hierarchical modules linked together via different types of communication systems. In order to control the complexity of distributed real-time systems, the following main requirements have to be fulfilled: reliable concept for decomposition and modularity, openness for extensions in the cases of new products, machines and devices, the general architecture model of the system has to be producer independent, use of encapsulated, reusable components.

The object-oriented approach ensures three main properties of the modelled system and its entities: encapsulation, inheritance and polymorphism. Encapsulation allows software to be developed on a modular principle, inheritance supports it's automatically reuse, thereby increasing robustness and significant facilitates the development processes. Polymorphism facilitates the development of independent modules in a constructive manner. The main disadvantages of object-oriented modelling are related to the absence of abstraction or support to the relationships and interactions between objects. Complex applications usually require interoperability between multiple objects. When designing and creating objects is not always known how and when to use their functionality and their state changes. This can lead to unexpected errors. Component-based development goes a step further by introducing more abstraction in terms of self-descriptive characteristics, which enable connections to other objects and usage of external functionality. This facilitates the development processes, but unfortunately does not contribute significantly the achievement of reuse and modification of the interface in real time, i.e. components used are also passive and can be described or used from outside. One of promising directions for achieving the above objectives in the development of control systems is the development of multi-agent systems by applying the principles of agent-oriented software engineering. Some of the most important features and principles of multi-agent systems are briefly discussed in the next part of the chapter.

3. Multi-agent systems

Multi-agent systems are characterized by local autonomy, social interaction, adaptability, robustness and scalability, and for these reasons are a very promising paradigm for addressing the challenges facing automation and control systems, associated with the development, operation and maintenance of reconfigurable distributed control systems. On the one hand multi-agent systems are widely accepted research topic in the field of automation and control systems, on the other, however, the field of automation and control is a major challenge to the agent-oriented software engineering to revealing its power, efficiency and relevance. MAS can be defined as "a loosely coupled network of problem solvers (agents) that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" [Durfee & Lesser, 1998]. Jennings and Wooldridge [Jennings & Wooldridge 1998] have defined an agent as "a computer system situated in some environment and capable of autonomous action in this environment, in order to meet its design objectives". Agents have the following main properties and characteristics [Wooldridge & Jennings, 1995]:

- autonomy: agents encapsulate some state (that is not accessible to other agents), and make decisions about what to do based on this state, without the direct intervention of humans or others;
- socialability: agents interact with other agents (and possibly humans) via some kind of agent-communication language, and typically have the ability to engage in social activities (such as cooperative problem solving or negotiation) in order to achieve their goals.
- reactivity: agents are situated in an environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps many of these combined), are able to perceive this environment (through the use of potentially imperfect sensors), and are able to respond in a timely fashion to changes that occur in it;
- pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative;

Opportunities for modelling of multi-agent systems at a high level of abstraction is a research topic in the field of Agent-Oriented Software Engineering (AOSE), which is developing very actively over the last few years. The research community in the field of AOSE investigates the UML extensions to support the modelling of basic agent-oriented concepts such as agents, ontology and interaction protocols. From a software engineering perspective, agent systems are a specialization of object-based systems, in which individual objects have their own threads of control and their own goals or sense of purpose. With the development of UML and the introduction of active objects, the differences between objects and agents decreased, owing to the improvement of the language meta-model and the establishment of new notations and formalization of the various diagrams used in the language. This allows achieving a higher degree of coverage of the basic properties of the agents and their interactions. Moreover, major advantage of using UML in the development of MAS compared with other alternatives is that the use of standard mechanisms and specifications is critical to ensure interoperability between autonomous agents in an open and distributed, agent-based integrated control environment.

Use of UML for development of multi-agent control systems beyond the problems with MAS, raises other important issues too. They relate to the satisfaction of real-time

requirements to the control systems, their quantitative assessments, as well as the procedures for verification and validation of their behaviour. The next section gives an overview and analysis of some different approaches to solving these problems.

4. UML extensions for modelling of multi-agent based control systems

4.1 Overview of real time extensions to UML

UML is a general purpose modelling language, which gives uniform notations and a meta-model for object-oriented modelling and software design. It does not specify a methodology for software design but supports the Model Driven Architecture (MDA) consisting in transformation of different platform independent models towards executable applications. UML aims to provide an integrated modelling framework, covering structural, functional and behaviour descriptions.

The main challenges towards a real-time UML are connected with the creation of different mechanisms to handle real-time features such as: models of physical time, timing specifications, timing facilities, modelling and management of physical resources and concurrency. In the last years there are many working groups developing and using different approaches and techniques in order to extend the object-oriented software development and especially UML to the field of real time system modelling and analysis. Further in the next paragraphs we try to summarize and analyze these research activities and the current state of the art in order to select and use the most suitable and effective approach for design and analysis of multi-agent based distributed control systems.

The first attempts to model control systems using UML appear almost parallel with the emergence of UML, setting some guidelines for the development of language. There are many different proposals for extending UML to support the design and analysis of real-time systems and they can be joined in 2 main abstract classes, shown in Fig.1. The first class of extensions follows the idea of Douglas [Douglass, 1998] and Selic [Selic, 1998] that the behaviour of complex real-time control systems can be fully described by using the standard capabilities of UML and is subdivided in two classes applying the following extension strategies:

- Changing the UML meta-model by explicitly adding new meta-classes and other meta-constructs;
- Creating UML profiles (standard and specific) on the base of stereotypes, constraints and tagged values. The 3 built-in extension mechanisms can be used separately or together.

Extensions to the first subclass are a priority of OMG (Object Management Group), but there are other initiatives too as for example the UML+, where new elements are added to the UML metamodel in order to achieve RT capabilities [Lavazza et al., 2001], [Bianco et al., 2001], [Bianco et al., 2002]. The current work on the evolution of UML standard at OMG aims the integration of the most successfully contributions to the real time issues. UML2.x provides means for architectural modelling inspired from UML-RT [Gerard et al., 2002], with structured classes, ports (isolate an object internals from its environment), connectors (which link communicating ports) and protocols (defined, reusable, interaction sequences). A structured class may have an internal structure. This structure is composed of structured classes, ports, protocols, etc. The idea of hierarchy appears too. UML 2.x presents some features that support real-time aspects and they may be summarized as follows:

- Support the modelling of concurrency through introduction of active objects, concurrent composite states and concurrent operations.

- For expressing of timing constraints two new data types are provided – Time and Time Expression, which may be used in state and sequence Diagram.
- Introducing of new diagram, so called Timing Diagram for expressing the timing behaviour of the system.
- Better semantic definition of the state diagram.
- Changes in the activity diagram, inspired from the advantages of Petri nets.
- New graphical features for the class diagram and new notations for the existing diagrams.
- Better definition of components and subsystems.
- Improved extension mechanism in respect to the user-defined meta-classes and relations, and profile mechanisms.
- UML 2.x is more open to formalization and includes more run time semantics.

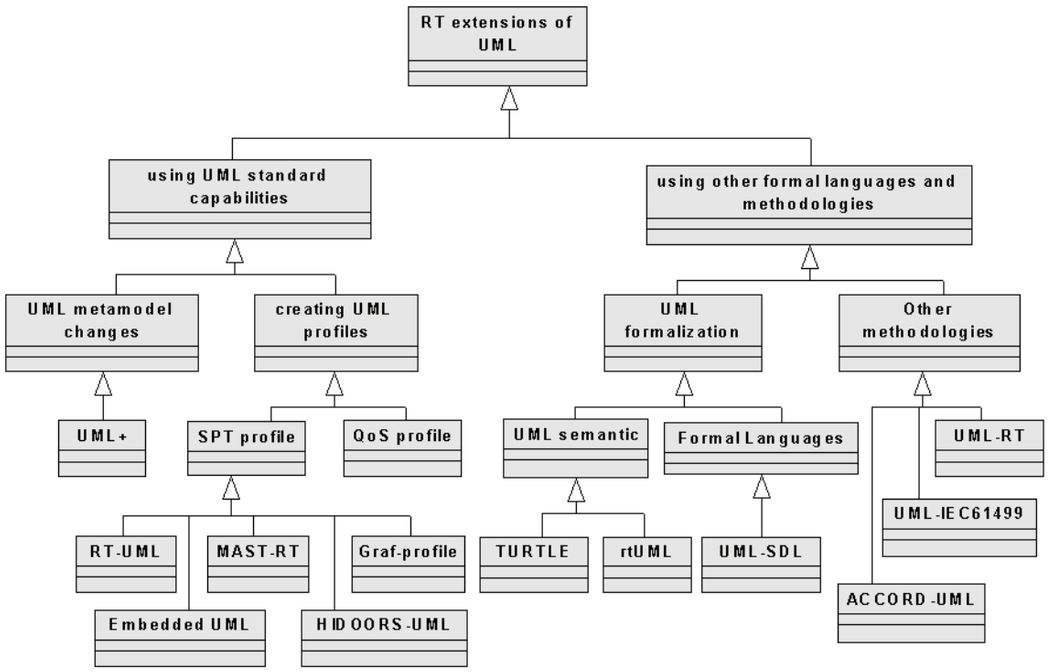


Fig. 1. UML extensions for real time

Despite the variety of structural and behaviour improvements, the new versions of UML2.x has still insufficiently support for development of distributed control system in respect to expressing hardware-software interdependences and timing models. That is why the using of specialized profiles based on UML2.x is the best solution for these purposes.

The objective of UML profiles, which are the focus of the second subclass of approaches, is to package terminology and substructures, specific for a particular application domain. One of the first attempt to provide RT capabilities of UML in this direction is the OMG initiative for creating of the profile for Schedulability, Performance, and Time Specification (SPT-profile) that is proposing a framework to model quality of service, resource, time and concurrency concepts in order to support predictive quantitative analysis of the UML models [OMG, 2003]. This profile supports two well-established forms of time-based model analysis: schedulability analysis based on schedulability theory and performance analysis

based on queuing theory or Stochastic Petri Nets. The SPT-profile is used as a basis for some other UML profiles some of them are RT-UML [DiPippo, 2000], MAST-RT [Medina et al., 2001], embedded UML [Grant&Jean, 2001], HIDOORS UML profile [Kesten, 1992], Graf-Ober profile [Graf&Ober, 2003] etc. An implementation of this idea is the CASE tool of I-logix – Rhapsody that will be used to illustrate the proposed approach in this chapter. The new UML MARTE profile supporting the specification, design and verification/validation in the model driven development of real time and embedded systems will replace the SPT profile with the appearance of the new UML2.3 version [OMG-UML, 2010]. UML profile for modelling Quality of Service and fault tolerance characteristics and mechanisms (QoS-profile) adopted by OMG in 2004 supports the implementation of SPT-profile through the definition of a catalogue with different categories of QoS characteristics, Quality models and UML model annotations using QoS requirements [Miguel, 2003].

The second class of RT extensions is connected with the combination of standard UML capabilities with those of other real-time frameworks, languages and methods, covering different real time features of the designed systems.

One very large class of RT extensions of UML is connected with the filling the gap from the lack of well-defined formal semantic and ensuring of capabilities for formal verification of designed UML models. In general, these extensions can be separated into two major categories. First of them is based on defining of the semantic domain and giving a sound semantics to the graphical notations in this domain. Examples of such profiles are TURTLE [Apvrille et al., 2004] and rtUML (krtUML) [Damm et al., 2002]. The plentiful graphical notations however make the semantic domain very complex, and impede the analysis tool support. This obstacle gives an advantage to the more workable approach including a combination of graphical notations and formal specification languages, and exploring the formal reasoning capabilities of formal methods. There is variety of works in the field of software engineering based on well-established traditional formal methods such as VDM, Z, B, and their object-oriented extensions such as VDM++, Z++, Object-Z etc. [Mwaluseke&Bowen, 2001]. One of the mostly used combinations in the development of RT applications is that between UML and SDL, which share a number of qualities, like having a graphical notation, good readability and good tool support. They also incorporate object orientation and state machines, which make UML and SDL suitable to work together [Verschaeve, 1999].

Typical representative of the approaches, which combine UML with other methodologies, is the UML-RT profile of IBM Rational Rose. It is based on some concepts of the ObjecTime ROOM methodology and provides three principal constructs (capsules, ports, and connectors) for modelling the structures of a real-time system using the basic UML1.4 mechanisms of stereotypes and tagged values. The main shortcomings of UML-RT are in specification of time constrains and timeliness properties because there are not means for it. It is more suitable for modelling the structure and communications between the different elements in the system.

Another very promising approach especially applied for development of distributed real-time systems in control and automation domain is the collaborative use of UML and the models proposed in the IEC-61499 standard. A short overview of the proposals in this field are presented and analyzed in part 4.3 of the chapter.

The main conclusions, which are deduced from the analyzed approaches for real time extensions of UML, may be summarized as follows:

- There is a strong interest worldwide on the real-time extensions of UML and their application in different applications domain;

- Current proposals for real-time UML are not complete and entirely compatible;
- Most of the proposals provide a good support for modelling of concurrent processes but are meagre to express quantitative real-time features (such as deadlines, periods, priorities);
- There are many proposals based on proprietary solutions, which are not fully compliant with the UML standard.

The main disadvantages of UML2.1 in respect to solving the tasks of control and system engineering are:

- Inability to model requirements and parametric equations and linking them to the structure class diagrams;
- Inability to model flows in the structure class diagrams;
- Inability to model hierarchical structures in detail.

4.2 UML profile for system engineering - SysML

SysML is a general-purpose modelling language for system engineering that reuses a subset of the last UML2.1 version and provides additional extensions through stereotypes, diagram extensions and model library in order to model a wide range of system engineering problems as for example specifying requirements, structure, behaviour, allocations and constraints on system properties to support engineering analysis. The reusable subset of UML, known as UML4SysML includes Interactions, State machines, Use Cases and Profiles. In Fig.2 the set of SysML diagrams in respect to their modelling aspects is summarized.

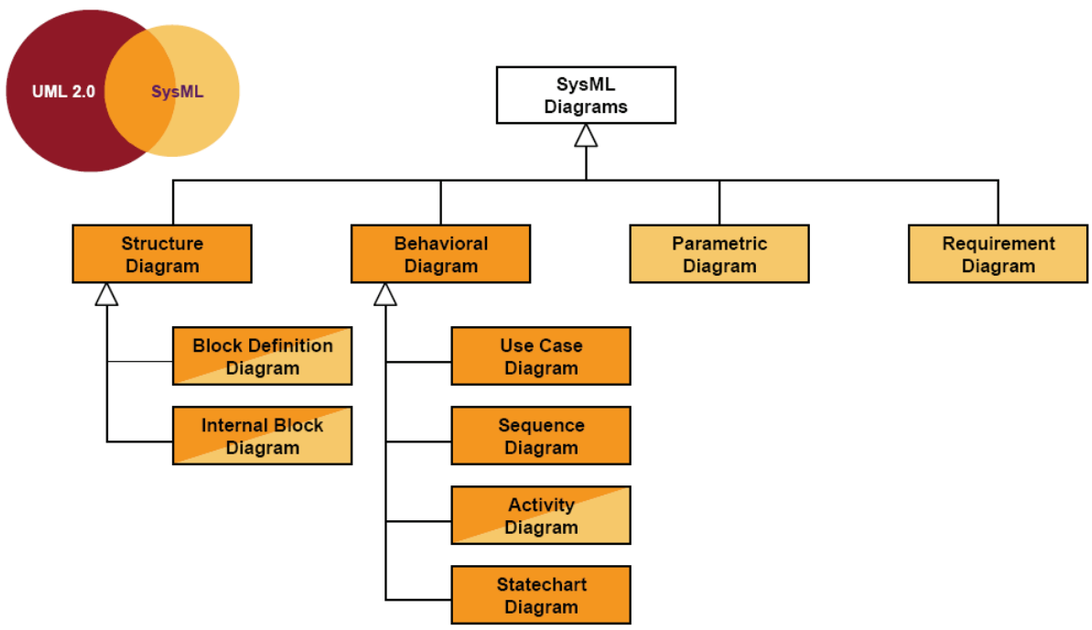


Fig. 2. SysML Diagrams [OMG-SysML, 2006]

The system structure design is supported by four types of diagrams: Block Definition Diagram (BDD), Internal Block Diagram (IBD) reinforced by Parametric and Packages diagrams. The BDD is an extended and restricted UML class diagram that is composed of blocks and relationships between them, such as associations, generalizations and dependences. The SysML blocks are based on UML classes with extended composite

structure and are defined in terms of properties, operations and relationships. The IBD captures the internal structure of a block in terms of properties and connectors between properties. Three general categories of properties are recognized: values, parts and references to other blocks. Ports are a special kind of parts, which give access to internal structures from the outside of a composite object. In SysML, unlike UML, ports can also be complex hierarchical structures. There are two basic types of ports – standard and flow ports. Standard ports are the same as in UML2.x and allow specifying services to or from the environment through the definition of provided and required interfaces. Flow ports are an extension of UML2.x ports and specify what "can" flow in or out of a block. Different items such as data, material or energy may flow through these interaction points in order to achieve asynchronous, broadcast, or send and forget interactions. The concept of item flows specifies what "does" flow between blocks and/or parts and across associations or connectors in a particular usage context. Parametric diagrams allow capturing system properties and constraints in form of different equations (algebraic, logical, differential, etc.) and support the analysis and verification processes. Packages diagrams are the same as in UML2.x and facilitate the decomposition and encapsulation of the projects.

The Behaviour diagrams incorporate four diagrams too, namely: activity diagram, sequence diagram, state machine diagram, and use case diagram. The activity diagrams are used to describe the flow of control and flow of inputs and outputs among actions. The state machines describe the constructs used to specify state based run time behaviour in terms of system states and their transitions. The sequence diagrams describe the communications between different structural elements over the time and identify the required relationships and messages. The use case diagrams model the relationships between users and a system enabling the specification of requirements to the system behaviour at the different decomposition levels.

The Requirements diagrams, which can be presented in graphical, tabular or tree structure format, are used to specify different constructs for system requirements and to cover the relationships between them. In SysML two kinds of requirements are used – functional and performance, as they specify the capabilities or the conditions which must be performed or satisfied by the system.

Other modelling capabilities of SysML, not shown in fig.2 are the cross-cutting constructs, such as allocations for connecting the different views, and Profiles & Model libraries allowing further to customize and extend SysML to specific applications. SysML includes extensions supporting the causal analysis, the verification and testing processes and the decision tree development.

SysML is developed as an open source specification and the last draft SysML1.2 is submitted to the OMG for technology adoption in June 2010. There are numerous modelling tool vendors who have already updated their UML2.x based tools to comply with the OMG SysML specification as for example ARTiSAN Studio, Rhapsody, MagicDraw, Enterprise Architect etc. [Huynh&Osmundson, 2005]. For the case study, described in this chapter Rhapsody of Telelogic is used [Rhapsody].

The advantages on the UML profile for system engineering SysML in comparison to these based on UML2.x may be summarized as follow:

- SysML supports the whole life cycle by the development of control engineering applications from the requirements definition to the software implementation;
- Based on extended activity diagrams, parametric diagrams and flow ports and items, the proposed approach may be applied for continuous and hybrid systems;

- The possibilities to model the physical systems in detail enhance the procedures for analysis, testing and validation of designed closed loop behavior of the system;
- Through encapsulated objects in UML 2.x and hierarchical structure, which they gain using SysML profile the object-oriented systems get near to agent-based systems;
- In order to improve the modelling processes, the UML/SysML notations may be mapped to concepts from IEC-61499 standard.

4.3 Development of IEC-61499 based control systems using UML

The IEC-61499 standard defines the basic concepts, reference models and architecture for development of modular, re-usable and open, vendor independent distributed process measurement and control applications, characterized through three main features: interoperability, portability and configurability [IEC-61499, 2005]. This standard uses the function block (FB) concept as a main building block of a control system, represented at different level of integration – device, resource and application. Three different kinds of FB are defined: basic (BFB), composite (CFB) for encapsulation of complex functionality through networks of BFB and service interface function block (SIFB) for providing interfaces for unidirectional (publish/subscribe) and bi-directional (client/server) communications. The control system may be modelled as logically connected function blocks through their input and output data and events. The BFB, shown in Fig.3 is presented by an input and an output interface composed of input and output events and data. The internal view of a basic function block includes an Execution Control Chart (ECC), internal data and internal algorithms. The ECC as shown in Fig.4 is a state machine, consisting of states, transitions and actions, invoking the execution of algorithms, associated to ECC states, in response to event inputs. One of the states is initial state and all other execution control states may have one algorithm and/or one output event associated. The evolution of the ECC state machine from an execution control state to other is realized by the execution of control transitions. In general the ECC is the relationship between events and algorithm executions, which are specified by the special kind of even-driven state machines. Every function block is characterized by its type name and instance name, which are used to identify a function block.

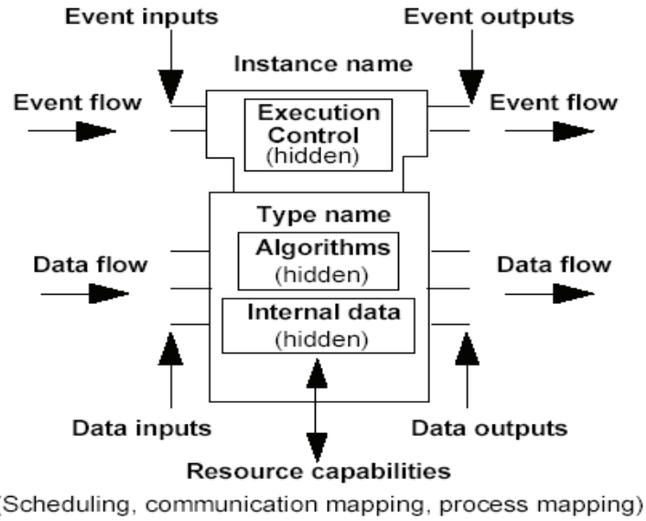


Fig. 3. Structure of a basic function block (IEC-61499)

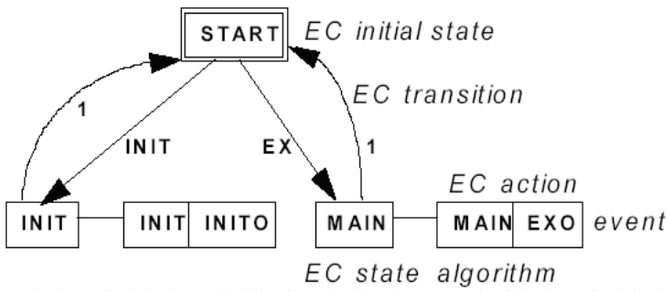


Fig. 4. Basic elements of the Execution Control Chart

Some of the main disadvantages of IEC-61499 standard are the absence of unified methodology for development of IEC-61499 based control systems and of formal semantic for describing control applications. This means that the approach based on the function block concept do not take into account the current situation and practice in the field of software engineering and makes the development processes and validation/verification phases of designed control systems difficult.

There are mainly two methodologies, which try to overcome these shortcomings using both UML and IEC-61499 standard for development of distributed control systems. The first approach, suggested by Thramboulidis, describes the model structure using class and sequence diagrams [Thramboulidis, 2001]. For these purposes, different stereotypes in respect to the IEC-61499 standard are defined as the FB stereotype, IPT (Industrial Process Terminator), the Data- and Control Dependency stereotypes, and Real-World Object Dependency stereotype [Thramboulidis, 2004]. The second approach suggests the control systems to be designed in an UML environment and after that the model to be transformed by well-defined transformation rules in FB based model using FBDK [FBDK, 2010]. A new modelling language UML-FB is defined [Dubinin&Vyatkin, 2004].

The approach, suggested in this chapter overcomes the shortcoming of the purely functional IEC-61499 standard by applying combined methodology based on the benefits of SysML and IEC-61499 standard in order to extend the standardized concept to the whole development life cycle of a control application, to model the close loop system and to use all benefits of object-oriented paradigm. There is a clear correspondence between the UML profile for System engineering SysML and IEC-61499 based models. The modelling concepts in both standards share many similarities. For example FB and parts are analogous, IEC-61499 data and event interfaces and ports are similar. `

4.4 Modelling MAS control systems using UML/SysML

The agent community has considerable interest in developing methods and techniques for specifying, modelling, implementing and verifying of multi-agent systems (MAS) for distributed control, but so far no standardized design methodology has been recognized. The successful industrial deployment of agent technology promotes techniques that reduce the risk inherent in any new technology, i.e. using standard representation of methods and tools. The successfully mission of UML as unification of numerous different object-oriented approaches led to the idea of applying UML to the design of MAS and it is widely applied for the purposes of agent-oriented software engineering. From a software point of view, agent-based systems can be regarded as a specialization of object-oriented systems but UML does not provide enough means for capturing in fully degree all agents related modelling aspects like autonomy, pro-activity and cooperation. The software agents have their own

thread of control, localizing not only code and state but their invocations as well. The UML's notations are extended to reflect the characteristic properties of the agents. OMG and FIPA (FIPA, 2010), (Bauer, 1999) are aiming to increase the acceptance of agent-based technology in industry by relating to de-facto standards for object-oriented software development and supporting the development environment throughout the full system lifecycle.
There are different extensions of UML trying to extend different aspects and in different manner the object-oriented modelling language. For example MAS-ML extends UML on the base of TAO (Taming Agents and Objects) meta-model (Silva et al., 2004), Different often used methodologies for development of MAS are supported by UML and AUML, such as for example Tropos, SABPO, Prometheus, MESSAGE, MaSE, PASSI and GAIA (only conceptually). A short overview of the most of them is given in [Mellouli et al., 2004], [Dam & Winikoff, 2003]. In [Sudeikat et al.] a genealogy of the most popular MAS methodologies is proposed in order to examine the gap between modelling and platform.
The mostly used and popular extension of UML for specification of agent-based systems is called AUML (Agent UML). AUML attempts to extend UML with agent-specific concepts and notations in order to model agents and interactions between them. The most significant are connected with the development of Agent Interaction Protocol (AIP) and are in the field of sequence diagrams, interaction overview diagrams, communication diagrams and timing diagrams. Interactions as key component of MAS are represented by set of rules known as interaction protocol, based on Speech Act Theory using verbs and contents [FIPA Modelling TC, 2003]. The communication protocols, which are brightly applied in distributed systems, are not suitable for describing the interactions in MAS, because of their process orientation [Lind, 2002]. The definition of interaction protocols is part of the specification of the dynamical model of an agent-based system. In UML, this model is captured by interaction diagrams (sequence and collaboration diagrams), statechart and activity diagram [Odell et al., 2000]. While the sequence diagram is one of the most commonly used for this purpose diagrams, then the statechart is not commonly used to express interaction protocols because it is a state-centric view, rather than an agent- or process-centred view. Our approach uses the activity diagram in order to work with specifications with very clear processing - thread semantics [Odell et al., 2000], [Bauer et al., 2004]. The Activity Diagram differs from the Interaction Diagrams because it provides an explicit thread of control. It is particularly useful for complex interaction protocols that involve concurrent processing. In UML2.x Activity Diagram, agents can be allocated to one dimension and locations are represented by orthogonal dimension so that the behaviour of multi-agents are easily captured and visualized. Signal sending and Signal receipt in UML Activity Diagrams enables agents to exchange messages, and synchronization of messages is achieved only when the parameters (message description) are exactly the same between signal sending and expecting signal receipt. Communications between classes are described by Agent Communication Language (ACL) messages using the pre-defined stereotype <<ACLMessage>>.
Based on the basic standards, methods and tools described and analysed in this and the previous two parts as well as their advantages and disadvantages for the development of advanced multi-agent based control systems, in the next part of the chapter the core idea and most important steps of the proposed approach are briefly presented.

5. Shortly description of suggested approach

The approaches in the area of distributed control systems existing till this moment are based only on UML/SysML or on the combined use of UML and IEC-61499 standard, as described

above. In the approach using UML/SysML profile the SysML stereotypes define new modelling constructs by customizing the existing UML constructs with new properties and constructs in order to create a framework which is a dynamic evolutionary environment, providing traceability and consistency. A SysML specification would be a much better start for the system development than a specification in natural language. But there is a fundamental difference between UML and SysML in the sense that UML models for software systems are intended to employ the same concepts during the complete development phases, reflecting the final software. Through encapsulated objects in UML 2.x and hierarchical structure which they gain using SysML profile the OO systems get near to agent-based systems.

SysML like UML is not a methodology and there is a need to choose or suggest an approach, which to be applied for the development processes. There are some successfully applied Model Based Systems Engineering (MBSE) methodologies such as Harmony SE, INCOSE Object-Oriented System Engineering (OOSE) Methodology, and IBM Rational Unified Process for SE (RUP SE), Vitech methodology, State Analysis (SA) methodology etc. A survey of these methodologies is given in [Estefan, 2007]. The proposed methodology is based on Harmony-SE, which is a subset of a large methodology for integrated system and software development process [Hoffman, 2006]. The task flow development process includes the following basic cycles as shown in Fig.5: Requirements Analysis; System functional analysis; Architecture design; Hardware/Software design specification.

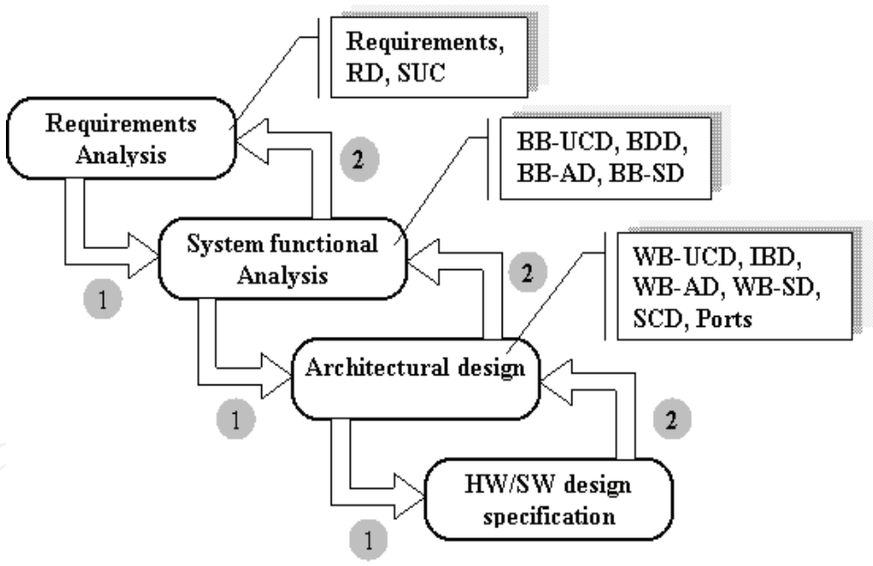


Fig. 5. Harmony SE based MBSE methodology

The proposed methodology uses service-request-driven modelling approach based on SysML structure diagrams using blocks as basic structural elements. The requirements analysis phase starts with the analysis of process inputs. Customer requirements are translated into a set of requirements that define what the system must do and how well it must perform. For the purposes of this first stage the SysML requirements diagrams (RD) to create taxonomy of the captured requirements and System Use Cases (SUC) are used. The system functional analysis phase is presented through transformation on the identified functional requirements into coherent system functions. Use Case Diagram (UCD) present

the system functional phase. For each use case the analysis is performed. BDD and IBD are applied to present the composite system structure. Different Black-Box Activity (BB-AD) and Sequence diagrams (BB-SD) are used to capture the high level system functionality. The Architecture design is the third stage in the proposed methodology and includes the design of subsystem structures and behaviour based on White-Box (WB) variants of the diagrams used in the previous stage (WB-UCD, WB-AD, WB-SD). Other important tasks are the ports and interfaces definitions and the description of subsystem state-based behaviour using the SysML statecharts (SCD). The last stage is the Hardware/Software design specification and is closely connected with the system implementation. All methodology stages include verification and validation tasks, based on the model developed in the previous stages and the defined requirements.

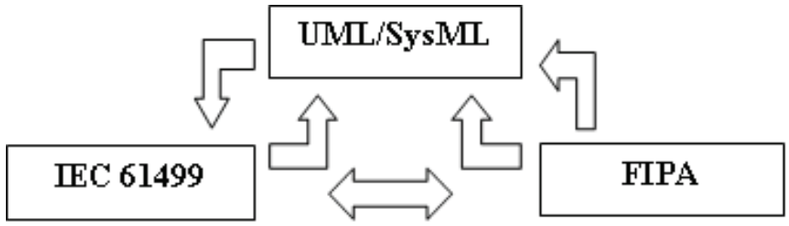


Fig. 6. Research framework

The methodology described above is used to model the internal structure of control system and to define how this system can be mapped to physical entities. However the standards used aren't enough to model the communications between Human Machine Interface (HMI) and control systems. For this reason the FIPA reference model is suitable to be used. The proposed research framework is shown in Fig. 6.

The FIPA agent reference model was chosen in order to provide a normative framework within which agents can be deployed and operate. FIPA specification establishes the logical reference model for creation, registration, location, communication, migration and retirement of agents. FIPA standard do not attempt to prescribe the internal architecture of agents nor how they should be implemented, but it specifies the interfaces necessary to support interoperability between agent systems. The FIPA modelling standard for development of multi-agent systems uses Activity Diagram as part of AUML. According Lind, we adopted the SysML Activity Diagram to model agent interaction protocol without introducing any new modelling elements [Lind, 2002]. The approach uses existing UML concepts only and requires no additional modelling elements, which makes it easy for UML users to understand the notation without learning a completely new type of diagram. For this case, to specify the agent communication protocols, two predefined stereotypes: <<ACLMessage>> and <<role>> are used.

6. Case study

The suggested approach will be illustrated in this part with a simple example of distributed control system for Distribution Station that is a part of didactic test bench, manufactured by FESTO Corp. and is located at the Martin Luther University of Halle-Wittenberg. The Distribution Station, shown in Fig. 7, separates work pieces from a magazine and then feeds them for processing. The Distribution Station is decomposed into two modules - the feed magazine, known as Storage module and the transfer module. The Storage module

separates work pieces from the magazine by a double acting cylinder, which pushes out the bottom work piece from the gravity feed magazine. The end pusher positions and the filling level of the magazine are monitored by means of binary sensors. The magazine might be fed with work pieces in any order. The Transfer Module is a pneumatic handling device. The work pieces are picked by a vacuum suction cap and transferred in a range of 0^o to 180^o by a swivel drive. The available sensors and actuators for both modules are shown in Table 1. The distribution station is controlled by an individual controller, connected to the controller of another stations via an I/O interface. The distribution station contains console with some buttons, such as "Start", "Stop", "Reset" and "Ack" for achieving different operation modes of the station through operator interventions. Both mechatronic modules are controlled separately by distributed controllers that communicate with each other. The scenario for distributed control follows the idea, submitted in [Vyatkin et al., 2006] according to which the functioning of both controllers is based on the mutually exclusive admission to the end position of the feeder unit (from the Storage module), where work pieces are picked up by the Transfer module.

Fig. 7. View of the Distribution station

Inputs and outputs of the Feed Magazine module			
Outputs		Inputs	
DI 0.0	Pusher retracted	DO 0.0	Extend pusher
DI 0.1	Pusher extended		
DI 0.6	Magazine empty		
Inputs and outputs of the Transfer module			
Outputs		Inputs	
DI 0.2	Swivel drive at magazine	DO 0.1	Drive to pos. magazine
DI 0.3	Swivel drive at testing station	DO 0.2	Drive to pos. testing station
DI 0.5	Work pieces sucked in	DO 0.3	Vacuum off
		DO 0.4	Vacuum on

Table 1. Sensors and actuators of the feed magazine and transfer modules

The first two stages in the design of control system according to the suggested approach are the definition of system requirements through the Requirements Diagram (Fig.8-1) and system functionality definition using Use Case Diagram (Fig.8-2) that captures the interactions between the system and its users and the requirements refinements to the

distributed control application. The static composite structure of the system consisting of distributed controllers, console, distribution station and connections between them is presented through BDD diagram (Fig.8-5). It corresponds to a Subapplication (Application) from the IEC-61499 standard. The basic elements of this diagram are presented as "system blocks" and their analogue according IEC-61499 is the CFB. The "parts" nested in these system blocks correspond to BFB. The third development stage includes modelling of the internal structure of the system blocks included in the overall static structure using BDDs and IBDs. The internal structure of the station is presented as BDD that covers the system hierarchy described through composition relationships (Fig.8-4). The message exchange among the system blocks is presented by Sequence Diagrams as shown in Fig.8-6. The model view browser as planned according the suggested approach is presented in Fig.8-3.

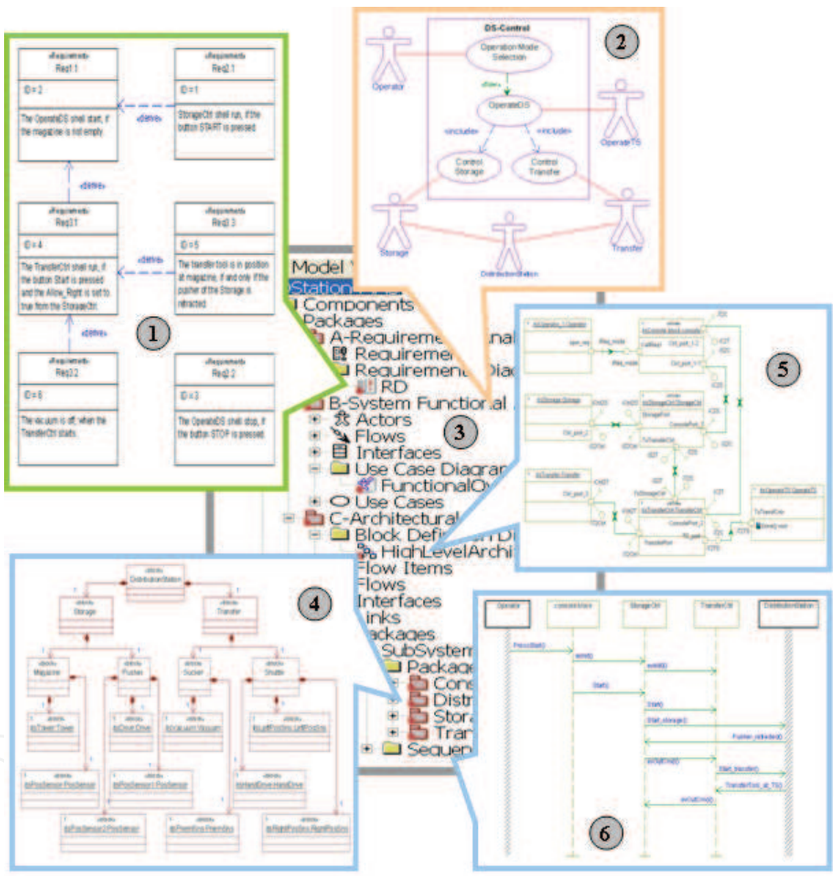


Fig. 8. Illustration of the first stages of the proposed approach

Both distributed controllers "StorageCtrl" and "TransferCtrl", modelled as "system blocks", are decomposed in two main "parts": "StorageCtrl_FB" and "E_Restart", and "TransferCtrl_FB" and "E_Restart" respectively. The controller internal structure covers the system hierarchy described through composition relationships and it is modelled through IBD as shown in Fig.9-1 for the "StorageCtrl" and in Fig.9-2 for the"TransferCtrl". Both parts in IBDs are modelled by parts with "BasicFunctionBlock" stereotype. The description of part "StorageCtrl_FB" includes the following attributes - *Empty, Pos_E, Pos_R, Retract, Left_OK, Right_OK, Allow_Left* and *Allow_Right* representing the input and output data and

operations, representing the input (*Init, Reset, Start, Sense, Stop, In_Cmd*) and output events (*Init0, Act* and *Out_Cmd*). The attributes of "TransferCtrl_FB" part are the following: *at_Mag, at_TS, Vac_on, Left_OK, Right_OK, to_Mag, to_TS, VCM_on, VCM_off*. The corresponding operations are the same and with the same meanings as for the "StorageCtrl_FB". The "E_Restart" part, used in both controllers, represents one restart event generator providing single events when the resource Cold and Warm starts and also when the resource is stopped by some external agents. It is likely that most applications will require at least one instance of this block to initiate the triggering event that starts execution of a chain of function blocks in a network.

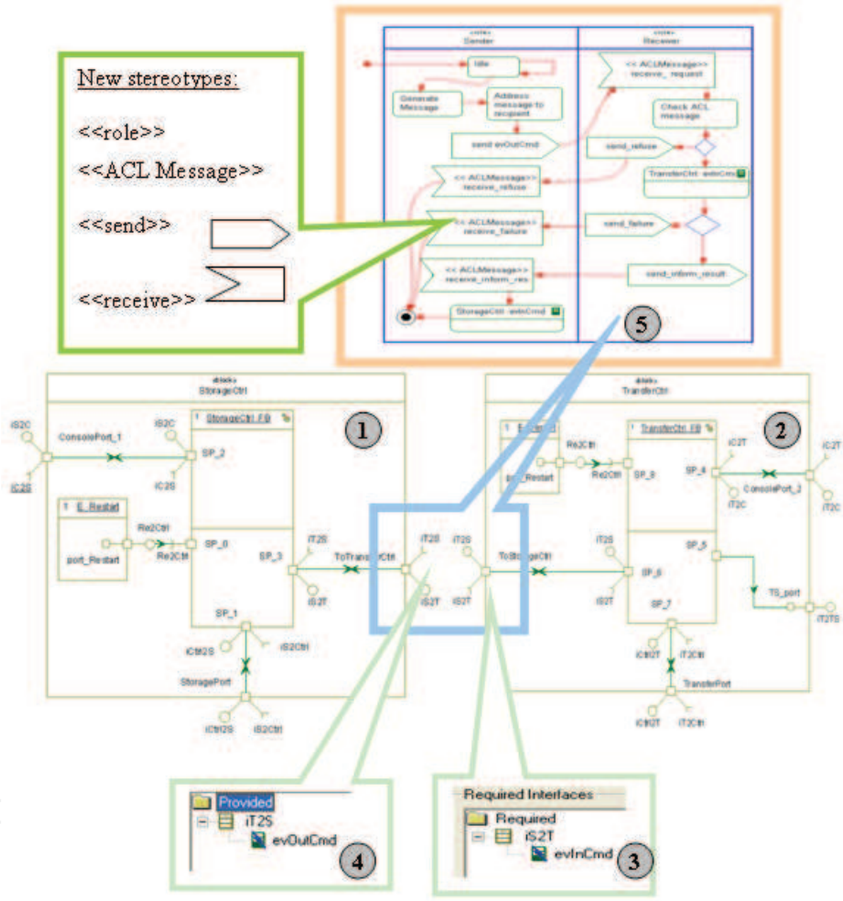


Fig. 9. IBD of the IEC-61499 based distributed controllers "StorageCtrl" and "TransferCtrl"

According to IEC-61499 standard, the link between function blocks and the interfaces of resources is provided through service interface FB (SIFB's). Communication interfaces in UML2.x can be defined by different types of ports. Contract-based ports are used in order to define contracts, which specify the precise allowable inputs and outputs of system components. The contract-based port between distributed controllers as shown in Fig.9 have two types of interfaces: provided (Fig.9-4) and required (Fig.9-3). The provided interface characterizes the requests from the environment and the required one – the request from the port. Because the ports are connected to other ports which are exactly "reversed" and with ports providing and requiring the same set of interfaces, the explicit contracts are used.

The FIPA "request when" protocol, as a kind of agreement for information exchange in a distributed system, can be used for peer-to-peer interactions between distributed controllers in order to obtain information or to request the execution of a task to other agents. This interaction protocol is modelled through activity diagram as behaviour of interface used and is illustrated in Fig.9-5. The diagram is customized through the introduction of stereotype <<role>>, associating swim lanes with roles. There are two roles in the FIPA "request when" protocol: sender and receiver, which are linked via implicit communication channels. In a protocol context the communication messages are grouped representing the sender – receiver direction. The protocol is used by the "sender" role of agent to request that "receiver" role performs some action at the time a given precondition becomes true (input event "evInCmd"). If an explicit agreement is required, the "receiver" can "refuse" to perform the action or "agree" to perform the action. If the "receiver" refuses, no further communication needs to take place. If the "receiver" agrees, it will attempt to perform the action when the precondition becomes true and then communicate to the "sender" one of the following: "failure" (the "receiver" failed to complete the requested action) or "inform-result" (the "receiver" has completed the action and notifies the "sender" of the result). According to FIPA reference model a class with stereotype "ACLMessage" is defined for representing of Agent Communication Language messages. Sending and receiving activities in this diagram allow agents to exchange messages based on communication protocols. The main benefits of using the FIPA Agent Interaction protocol is that it contains unified agreements on the methods used for initiation and termination of protocol data units (message), formatting and encoding data, synchronization of senders and receivers, and detection and correction of transmission errors. The protocol specification consists of the following parts: service specification, assumption about the environment, precise message format for (syntax), procedure rules for data exchange (grammar) and a vocabulary of messages used with their meaning (semantics).

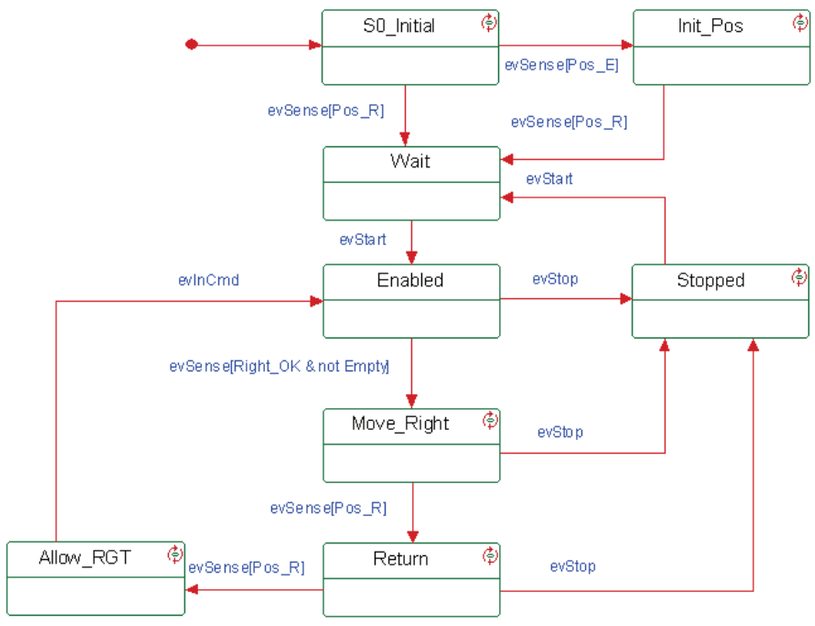


Fig. 10. Statechart of the "StorageCtrl_FB" system block

The system behaviour of basic FB according IEC-61499 standard is specified through Execution Control Chart (ECC). The functionality of ECC may be achieved for the system behaviour of IBD parts using SysML statecharts. The statecharts for both parts – "StorageCtrl_FB" and "TransferCtrl_FB" are shown in Fig.10 and Fig.11 respectively.
The statechart of part "StorageCtrl_FB" describes its basic states and the relations between them, modelled through the so called transitions. The control instance is in initial state when it is first created. From this state, after occurrence of input event "evInit", "StorageCtrl_FB" passes in initialization state "S0-Initial" where the controller memory is initialized and of the output data "Allow_Right" and "Allow_Left" is assigned value "false" in the frame of the algorithm "Alg_Init". From this state the object can move into two new states "Init_Pos" and "Wait". If the pusher is heading "back", i.e. the transition condition Sense[pos_R] is valid, control changes to state "Wait", which does not perform any actions. Otherwise, i.e. if the transition condition Sense[pos_E] is "true", the object "StorageCtrl_FB" goes into state "Init_Pos". In this state an action is initiated resulting in invoking of algorithm "Alg_Retract1" according that of the output data "retract" the value "true" is assigned (retract:= 1) and the output event "evAct" is generated (OUT_PORT (SP_1) -> GEN (evAct)), connected with activating the actuator to bring the pusher into position "retract". By satisfying the transition condition Sense[pos_R] (i.e. when the pusher is already in position "retract"), the object goes into a state of waiting "Wait". From that state the control object may be out after the occurrence of event "evStart", and than is able to go in the state "Enable", in which the object falls from the state "Allow_RGT" also, after the occurrence of "evInCmd" event. The control object can leave the state "Enable" in two ways: either after the occurrence of event "evStop", after that is passing in a state "Stopped", or by satisfying the guard condition [Right_OK & not Empty] (which means that the part can be manipulated by the swivel drive and the magazine is not empty) and than is passing in the state "Move_Right". In this state, on the base of algorithm "Alg_Retract0" the output data "retract" gets value "false", and on the base of "Alg_AllowRight0", the output data "Allow_Right" – the value "true" by generating output events "evAct" and "evOutCmd", respectively associated with the data. From this state, by occurrence of input event "evStop", the object falls in the already known state "Stopped", and by the occurrence of event "evSense" and guard condition [pos_E] (pusher is extended), the object passes in a state "Return", where the algorithm "Alg_Retract1" is running and the output data "retract" gets the value "true" after that the output event "evAct" is generated in order to return the pusher. As in previous states, and here by an occurrence of event "evStop", the object is passed to state "Stopped" where an algorithm "Alg_Retract1" (retract:=0) is executed and after that the output event "evAct" is generated. If an event "evSense" occurs in state "Return" and the guard condition [pos_E] is true (pusher is extended), the object passes into the state "Allow_Right", in which the variable "Allow_Right" gets value "true" as a result of the execution of "Alg_AllowRight1" algorithm and after that the output event "evOutCmd" is generated.
The statechart of part "TransferCtrl_FB" shown in Fig.11, starts from the initial state and after occurring of input event "evInit", the control object goes into state "Init". In this state the algorithm "Alg_Init" is invoked and after that an action on exit is executed, connected with the generating of output event "evInit0". From this state the object can move into two new states "Init_Pos" and "Enabled". If the swivel drive is not in position "at_Mag" and the left standing operation is completed, i.e. the guard condition [at_TS and Left_OK] is satisfied, the state "Init_Pos" is reached. In this state after the execution of algorithm

"Alg_ToMag1"(To_Mag:=1), an action "evAct/To_Mag:=1" is initiated in order to return the swivel drive in position at magazine. In this state a second algorithm "Alg_AckLed1" is invoked too and is connected with assignment of value "true" to the output data [Ack_led]. By satisfying the guard condition [at_Mag], the object goes into state "Enabled". From state "Init_Pos" the control object may be passed to state "Stopped" upon the occurrence of an event "evStop". In this state 3 algorithms are invoked: "Alg_ToTS0" (To_TS:=0), "Alg_ToMag0" (To_Mag:=0) and "Alg_StartLed1" (Start_led:=1). After the execution of the first of them, the output event "evAct"/To_TS:=0 is initiated. The state "Stopped" may be terminated by satisfying the guard condition [not Right_OK] after that the object is switched to state "Enabled" where the algorithm "Alg_AllowLeft0" (Allow_Left:=0) is executed. From state "Enabled", after pressing the button "Start" (trigger "evStart"), the state "Set_Led" is activated, where the [Right_OK] is not valid. In this state the algorithm "Alg_AckLed0" is running that assigns the value "false"to the output data [Ack_led] and generates the output event "evAct/Ack_led:=0". By satisfying the guard condition [Right_OK] the object enter the state "Move_TS", where after the execution of algorithm "Alg_ToTS1" (To_TS:=1), the output event "evAct" is initiated, that drives the transfer tool to position testing station. From this state the object can move into two alternative states: "Stopped" and "Hold". In state "Hold" the algorithms "Alg_AckLed0" (Ack_led:=0), "Alg_ToTS0" (To_TS:=0) and "Alg_AllowLeft1" (Allow_Left:=1") are executed, after that the actions "evAct/To_TS:=0" and "evOutCmd/Allow_Left:=1" are initiated in order to move the transfer tool to the Magazine and to enable the StorageCtrl_FB. From this state, by occurrence of input event "evStop", the object falls in the already known state "Stopped" and by occurrence of "evInCmd/Left_OK:=1" the object is passing in state "to_Mag" where two combinations of algorithms and actions are executed: the first one includes the algorithm "Alg_AllowLeft0" (Allow_Left:=0) and the action "evOutCmd/Allow_Left:=0", and the second "Alg_ToMag1" (to_Mag:=1) and "evAct/to_Mag:=1". By activating the transition condition [at_Mag:=1] the object is going to the next state "Suck_in", where the algorithm "Alg_VacOn1" (VCM_on:=1) is executed and the output event "evAct/VCM_on = true" is generated, where the work piece is picked by a vacuum suction cap. Another possible state is the "Stopped" state, appearing by pressing the button "Stop" (trigger "evStop"). If an event "evSense" occurs and the guard condition [Vac_on] is true, i.e. work piece is sucked in, the object passes into the state "To_TS" in which the algorithm "Alg_ToTS1" (To_TS:=1) is executed and the output event "evAct/To_TS:=1" is generated i.e. the swivel drive transfers the work piece to the Testing Station. After satisfying the guard condition [at_TS] the state "Drop" is achieved and the algorithm "Alg_VacOff" (VCM_off:=1) is executed and the output event "evAct/VCM_off:=1" is generated that is connected with release of the work piece. The state "Return_Mag" is reached from the state "Drop" in case that [Vac_on] = false and in this state the algorithm "Alg_ToMag1" (to_Mag:=1) is executed and output event "evAct/to_Mag:=1" is generated, which returns the swivel drive at magazine. If the event "evSense" occurs and the guard condition [at_Mag] is satisfied, the next state "Wait" is reached. In this state the algorithm "Alg_AckLed1" (Ack_Led:=1) is executed and after that the output event "evAct/ Ack_led:=1" is generated. A second algorithm "Alg_AllowRight1" (Allow_Right:=1) is performed too, after that the output event "evOutCmd/Allow_Right:=1" is initiated. From the state "Wait" by occurrence of events "evAck" or "evNotSingle" the control object is moving to state "Set_Led" and the module is ready to carry the same sequence of actions and related algorithms. The last four states include also transitions to state "Stopped" in the case of the occurrence of input event "evStop".

Algorithms which are executed in some states of the above described statecharts may be presented with Activity Diagrams. For this case study the algorithms used are very simple and there is not a need to model them.

7. Conclusions

The suggested methodology for development of multi-agent control systems based on the combined use of UML/SysML, FIPA and IEC-61499 standards is suitable for development of open, interoperable, re-configurable and distributed control system. This methodology allows describing the whole live-cycle of the control system and achieving software encapsulation and re-use of the defined entities (agents). One of the most essential features of this approach is that control engineers are able to model the closed loop control system and to apply the different type of analysis techniques in order to determine whether these models meet their performance and schedulability requirements, without requiring a deep understanding of the inner working of those techniques.

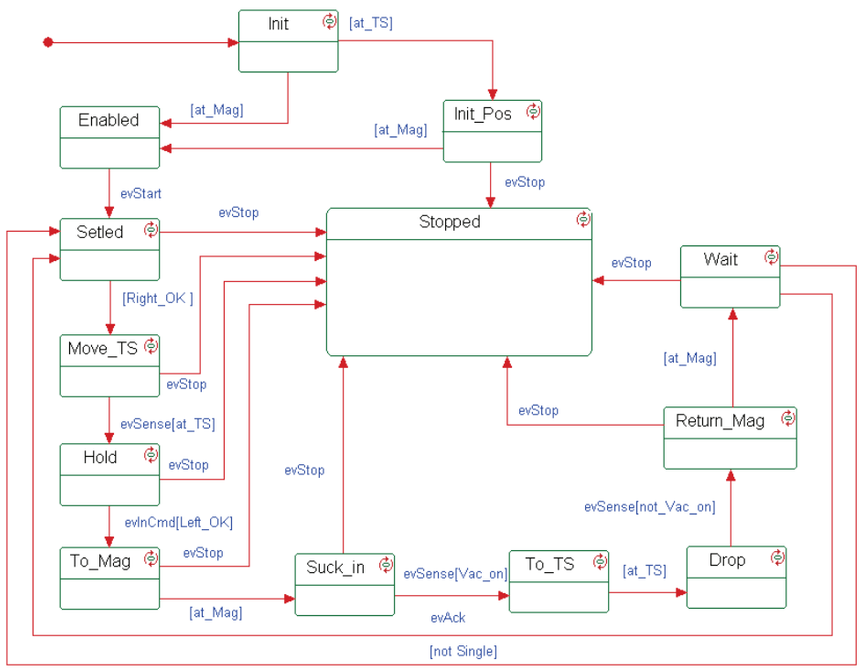


Fig. 11. Statechart of the "TransferCtrl-FB" part

The UML/SysML profile is extended and at the same time restricted with different stereotypes from the FIPA communication standard in order to fill the gap from absence of unified communication protocol in applications based on the IEC-61499.

The main advantages of the proposed approach may be summarized as follows:

- SysML supports the whole life cycle by the development of control engineering applications from the requirements definition to the software implementation;
- The possibilities to model the physical systems in detail enhance the procedures for analysis, testing and validation of designed closed loop behaviour of the system;
- Based on extended activity diagrams with FIPA standard, the proposed approach get near to agent-based systems.

8. Acknowledgment

This work has been partially supported by the Bulgarian National Research Fund to the Ministry of Education and Science in the frame of Project BУ-TH-208.

9. References

Apvrille, L., Courtiat, J.-P., Lohr, Ch. & Saqui-Sannes, P. (2004). TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit, *IEEE Transactions on Software Engineering*, Vol. 30, No. 7, July 2004, pp. 473-487

Bauer, B. (1999). Extending UML for the Specification of Interaction Protocols, submitted for *the 6th Call for Proposal of FIPA*.

Bauer, B., Muller, J. P. & Odell, J. (2001). Agent UML: A Formalism for Specifying Multi-agent Software Systems, *In Proceedings of the First International Workshop on Agent-Oriented Software Engineering AOSE'00*, Limerick, Ireland, pp. 91-103, LNCS 1957 Springer

Bianco, V., Lavazza, L. & Mauri, M. (2001). An introduction to the DESS approach to the specification of real-time software, *CEFRIEL Technical Report RT01002*, April

Bianco, V, Lavazza, L. & Mauri, M. (2002). A Formalization of UML Statecharts for Real-Time Software Modelling, *6th Biennial World Conference on Integrated Design Process Technology (IDPT2002), "Towards a rigorous UML"*, Pasadena, California, June 23-28.

Bussmann S., McFarlane D.C. (1999), Rationales for holonic manufacturing control, *Proc. of the 2nd International Workshop on Intelligent manufacturing Systems,* pp.177-184, September, Leuven, Belgium.

Damm, W., Josko, B., Pnueli, A. & Votintseva, A. (2002). Understanding UML: A Formal Semantics of Concurrency and Communication in Real-Time UML. *First International Symposium, FMCO 2002*, Leiden, The Netherlands, November 5-8, 2002, Revised Lectures, pp. 71-98

Dam, K. H. & Winikoff, M. (2003). Comparing Agent-Oriented Methodologies. in: *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems*, Melbourne, July (at AAMAS03).

DiPippo, L. C. (2000). A UML Package for Specifying Real-Time Objects, *Computer Standards & Interfaces*, Volume 22 , Issue 5 (December 2000) pp. 307–321,

Douglass, B. (1998). *Real-Time UML: Developing Efficient Objects for Embedded Systems*, Addison-Wesley Longman Publishing Co., Inc., ISBN:0201325799, Boston, MA,USA

Dubinin, V. & Vyatkin, V. (2004). UML-FB – A Language for Modelling and Implementation of Industrial-Process Measurement and Control Systems on the Basis of IEC 61499 Standard, *Proc. of the 6-th International Conference of Science and Technology "New Information Technologies and Systems (NITS'2004),* pp.77-83, Penza, Russia, Part 2, June 17-19

Durfee, E. H. & Lesser, V. (1989). Negotiating task decomposition and allocation using partial global planning, in: L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II.* Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989, pp. 229–244.

Estefan, J. A. (2007). Survey of model-based system engineering (MBSE) methodologies, pp.1-47, *INCOSE MBSE Focus Group,* May 25

FIPA (2010). Foundation for Intelligent Physical Agents, http://www.fipa.org

FIPA Modeling TC (2003). FIPA Modelling Area: Deployment and Mobility, *FIPA Technical report*. www.auml.org/auml/documents/DeploymentMobility.zip

Gerard, S., Terrier, F. & Tanguy, Y. (2002). Using the Model Paradigm for Real-Time Systems Development: ACCORD/UML, *Proc. Conf. Advances in Object-Oriented Information Systems, OOIS 2002 Workshops*, pp. 260-269, ISBN:3-540-44088-7, Springer-Verlag, London, UK

Graf, S. & Ober, I. (2003). A real-time profile for UML and how to adapt it to SDL, *In Proceedings of the 11th international conference on System design*, pp. 55-76, ISBN ~ ISSN:0302-9743, 3-540-40539-9 Springer Verlag, Berlin, Heidelberg: http://citeseer.ist.psu.edu/graf03realtime.html.

Hoffman, H.-P. (2001). Harmony-SE/SysML Deskbook Model-based Systems Engineering with Rhapsody, Rev.1.51, *Telelogic/I-Logix whitepaper*, May 24.

Huynh, T. & Osmundson, J. (2005). A System Engineering Methodology for Analysis Systems of Systems Using the System Modelling Language, Proceeding *of 1st Annual System of System Engineering Conference*, Johnstown, PA, June 13-14,

IEC-61499 (2005). *International Standard IEC-61499, Function Blocks, Part 1 - Part 4*, International Electrotechnical Commission (IEC), Technical Committee TC65/WG6, IEC Press, Jan.

Jennings N. R. & Wooldridge M. (1998). Applications of Agent Technology, in: N. R. Jennings and M. Wooldridge, editors, *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, March 1998.

Kesten, Y. (1992). Timed and Hybrid Statecharts and Their Textual Representation, Formal Techniques in Real-Time and Fault-Tolerant Systems, *2nd International Symposium,* pp. 591 – 620, ISBN: 3-540-55092-5, Springer-Verlag, London, UK

Lavazza L., Quaroni G. & Venturelli M. (2001). Combining UML and formal notations for modelling realtime systems. V. Gruhn (ed.) *Proceedings of ESEC/FSE 2001*, Vienna, Austria, September 10-14, ACM Press

Lind, J. (2002). Specifying Agent Interaction Protocols with Standard UML, *In 2nd International Workshop on Agent-Oriented Software Engineering AOSE'02*, pp. 136-147, LNCS 2222 Springer

Medina, J. L., Drake, J. M. & González, M. H. (2001). UML-MAST: Modelling and Analysis Methodology for Real-Time Systems Developed with UML CASE Tools, *Proceedings of the Euromicro Conference on Real-Time Systems*, Delft, The Netherlands, June

Mellouli, S.; Moulin, B. & Mineau, G. W. (2004), Towards a Modelling Methodology for Fault-Tolerant Multi-Agent Systems, in: *Informatica Journal* 28, pp.31-40.

Meunier, J.-N., Lippert, F. and Jadhav, R. (2003). RT modeling with UML for safety critical applications: the HIDOORS project example, Proceedings of SVERTS 2003, Co-located with UML 2003, San Francisco, CA, U.S.A., October.

Miguel, A. (2003). General Framework for the Description of QoS in UML. *In EEE Computer Society, editor, Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)*, pp 61–70, Hakodate, Hokkaido, Japan, May

Mwaluseke, G. W. & Bowen, J. P. (2001). UML Formalisation Literature Survey, 6 September,

Odell, J., Parunak, H. & Bauer, B. (2000). Extending UML for Agents, In Proceedings of the Agent-Oriented Information Systems Workshop - AOIS 2000, pp. 3-17, Eds., Austin

Odell, J.; Parunak, H. V. D. & Bauer, B. (2001). Representing Agent Interaction Protocols in UML, *Agent-Oriented Software Engineering*, Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pp.121–140.

OMG (2003). UML Profile for Performance, Schedulability and Time, *Specification OMG*,

OMG-UML (2010).Unified Modelling Language Specification, June: http://www.omg.org/

OMG-UML (2003). UML for Systems Engineering, Request for Proposals - RFP ad/2003-3-41, *Object Management Group*, Needham, September.

OMG-SysML (2006). The OMG Systems Modelling Language, May http://omgsysml.org/index.htm

Parunak, H. V. D. (1999). Practical and Industrial Application of Agent-Based Systems, in: *Multiagent Systems*, G. Weiss, ed., Cambridge: The MIT Press.

Rhapsody, http://modeling.telelogic.com/standards/

Selic, B. (1998). Using UML for Modeling Complex Real-Time Systems, *Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems*, pp. 250 – 260, ISBN 978-3-540-65075-1, Springer-Verlag, London, UK

Selic, B. (2002). *The Real-Time UML Standard: Definition and Application*, IEEE Computer Society, ISSN:1530-1591, Washington, DC, USA

Selic, B. (2004). Tutorial: An Overview of UML 2.0, *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, IBM Rational Software Canada

Silva, V.; Choren, R. & Lucena, C. (2004). A UML Based Approach for Modelling and Implementing Multi-Agent Systems. In: *Proceeding of the third International Conference on Autonomous Agents and Multi-Agents Systems*, USA.

Sudeikat, J.; Braubach, L.; Pokahr, A., & Lamersdorf W. (2004). Evaluation of Agent–Oriented Software Methodologies – Examination of the Gap between Modelling and Platform, in: *Agent-Oriented Software Engineering V, Fifth International Workshop AOSE*, pp.126-141.

Telelogic AB (2000). *User Guide Rhapsody*, Rhapsody Tool.

Thramboulidis, K. C. (2001). Using UML for the Development of Distributed Industrial Process Measurement and Control Systems, *IEEE Conference on Control Applications (CCA)*, September, Mexico.

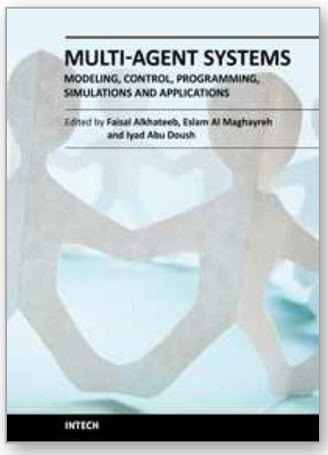
Thramboulidis, K. C. (2004). Using UML in Control and Automation: A Model Driven Approach, *2nd IEEE International Conference on Industrial Informatics INDIN'04*, 24th - 26th June, Berlin, Germany.

Parunak, H. V. D. (1999). Practical and Industrial Application of Agent-Based Systems, in: *Multiagent Systems*, G. Weiss, ed., Cambridge: The MIT Press.

Vyatkin, V., Hirsch, M. & Hanisch H.-M. (2006). Systematic Design and Implementation of Distributed Controllers in Industrial Automation, *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA´2006) proceedings*, pp.633-640, Prague, Czech Republic.

Verschaeve, K. (1999). Combining UML and SDL, *Eighth SDL Forum*, Montréal, Canada,

Wooldridge, M., Jennings, N. R. & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design, *International Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285-312.



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Iskra Antonova and Idilia Batchkova (2011). Development of Multi-Agent Control Systems using UML/SysML, Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-174-9, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-control-programming-simulations-and-applications/development-of-multi-agent-control-systems-using-uml-sysml

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen