

# TREND GUIDE

## Embedded UML



**MICRO CONSULT**

No. 1 in Developer Training



## UML für 16 Bit hilft.

Als Programmierer von Microcontrollern wissen Sie, wie komplex sich fortlaufende Änderungen auf eine Software auswirken können. Verlängern Sie die Lebensdauer Ihres Source Codes und vereinfachen Sie Ihre Arbeit mit:

## Embedded UML Studio16 Bit

OOP · UML · C/C++ · RTOS für C166/ST10/M16C/TriCore

UML CASE Tools

Echtzeitbetriebssystem

Testtool

Evaboards

C/C++-Compiler

Debugger

[www.willert.de](http://www.willert.de)

Willert Software Tools GmbH

Herminenstr. 17 b  
D-31675 Bückeburg  
Tel +49 5722 9678 60  
Fax +49 5722 9678 80  
[info@willert.de](mailto:info@willert.de)

## Embedded UML

Preface **04**

Trend Spots **06**

Look Ahead – Rethink now! **08**

Create a Basis – Pathfinder for UML **12**

Starting Aids – Strategy of Small Steps **16**

Experience – Economical of RAM and ROM **24**

Overview on UML Real-Time Case Tools **30**

Outlook- UML 2.0 for Embedded Software? **31**

At a Glance – 10 Practical Tips **36**

UML is Profitable – Reduce Costs, Increase Efficiency **39**

Phases to Project Success – UML Step by Step **40**

Info Pool – Book and Web Tips **42**

MicroConsult Services – Related Training Courses **46**

Imprint **52**

---

Your Partners for Embedded UML! **50**

---



**Klaus-Peter Rosenthal**  
MicroConsult  
k.rosenthal@microconsult.de

*Dear Readers,*

*This we have learned over the past decades: The growing complexity of embedded software can only be mastered on a high level of abstraction. But this is not the only reason for the impressive evolution of UML in complex projects. However, pressure is growing even for smaller embedded systems: customers demand higher quality, faster time-to-market, reusability and higher flexibility.*

*So the primary question today is not only if you can afford to invest in UML and the required know-how. Or did it ever happen to you that you had more time available in a subsequent project or that the basic conditions for innovations were more favorable than in your previous project? You had better ask yourself how much longer your current software development method will be efficient. I think that time is ripe for a new era of embedded software development. The name is UML. This Trend Guide describes first ways to get started in your company.*

A handwritten signature in black ink that reads "Klaus-Peter Rosenthal".



**Andreas Willert**  
Willert Software Tools  
awillert@willert.de

*Dear Readers,*

*Do you want to survive in the embedded software market or even capture new territory? It's easy - just double your development team. You surely have a nice financial cushion, right? If not, go for the cheap variant. Outsource programming to India, the Czech Republic or some other low-wage country. After all, offshore outsourcing is the talk of the town. Outsource your competences along with it. No problem, is it?*

*Granted, but sometimes you have to carry something to extremes to see clearly again. The only choice we have in Germany, the high-wage country: highly efficient developer teams. This Trend Guide shows you how to do it.*

A handwritten signature in black ink that reads "Andreas Willert".

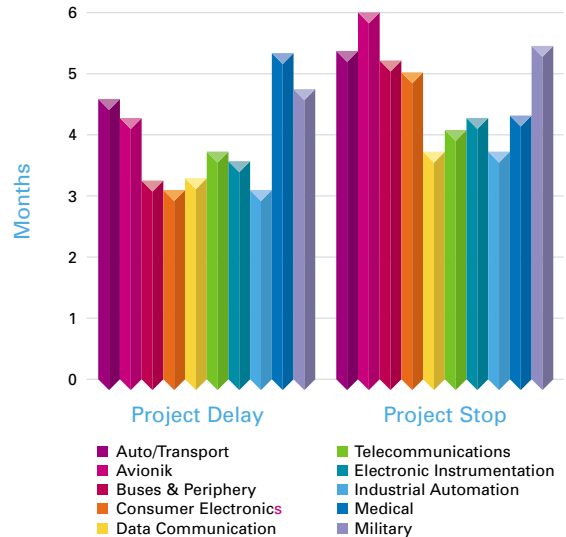
## A Short Summary

- Unnoticed by the broad public, embedded and automation software have turned into an important market. According to a recent study, every second industrial firm invests more than 30 percent of its development budget in software. But the study also shows the other side of the story. One third of the firms have already lost orders due to software shortcomings. *Source: Smartresearch 2003*
- “The problems that programmers are facing today are mainly based on the requirement for ever increasing functionality and capacity in real-time and embedded systems. At the same time, the human and time resources to develop these increasingly complex systems are reduced. With UML, developers can work with the latest state-of-the-art technologies to realize more powerful systems in a shorter time and with fewer errors.”  
*Bruce Powel Douglass, UML Guru and author of the bestseller “Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns”*

- In a current study, more than 33 percent of embedded developers stated that their final design does not even fulfill half of the initial requirements on performance, system functionality and time schedule. Their main way out of the crisis is extensive reengineering or the removal of functions.

*Source: Embedded Market Forecasters 2004*

### Longer than Expected



*At an average, projects take four months longer than planned because more than 50 percent of the embedded design is far behind the time schedule. In more than 80 percent of the cases, the reason lies in software development. Source: Embedded Market Forecasters 2004*

## Rethink now!

*Until only a few years ago, development companies had well filled order books, and good programmers were scarce in the market. Teams were built up extensively, regardless of the costs. At that time, hardly anybody thought seriously about “how” to develop software. Now this strategy gets back on the industry.*

Today, we have rapidly increasing complexity and shorter product cycles on the one hand, and tight budgets for labor on the other. And the situation will continue to aggravate, so that shortages cannot be compensated for with night and weekend work any longer. Even the end customer is now confronted with what happens behind the scenes. Just look at the endless series of callbacks in the automotive industry. The end of German high class workmanship seems near. At least for complex embedded software projects.

But is it really true that the teams are seriously understaffed, or does the situation rather reflect a drawback in software development? This question seems legitimate in view of the current trends. Software was actually meant to make hardware more flexible, promote innovation and quickly implement customer requirements. Now it turns out to be the strongest project costs booster in the high-wage country Germany.

In the meantime, the hardware industry has already recognized the signs of the time (see box) and clearly focuses on standardization. The main reason is probably that hardware generates higher profit – at clearly calculated production cost.



### Learning from the Hardware Industry

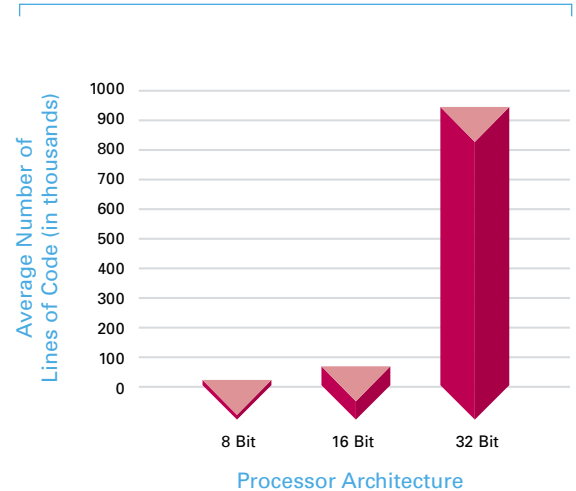
*About 15 years ago, hardware development engineers had to face a problem similar to that of embedded software today – complexity went out of control. Somewhere along the way, the costs for design changes were so high that the “try-and-error” method on a low level of abstraction (transistors, gates) was no longer profitable. Moreover, the long redesign time of several weeks became unacceptable. There was no way to get around a change of paradigm.*

*At that time, VHDL became the standard notation for describing hardware on a more abstract level. Developers could “debug” on this level by means of simulation and automatically generate the technology-specific implementation on gate level using logic synthesis. Without this innovation, suppliers would have never been in a position to develop chips with the complexity of a pentium processor under the given cost and time conditions. Just like hardware programming at that time, software programming now has to initiate a similar change of paradigm to free itself from the current dilemma. This commentary comes from Wolfgang Leimbach. The author has been working for leading EDA companies for more than 15 years, today as Sales Manager Central Europe for I-Logix. [www.ilogix.com](http://www.ilogix.com)*

Embedded software is traditionally supplied at no cost along with the hardware. To the customer, this value is not really measurable, because it is not really tangible in the truest sense of the word – even though today it already has the lion’s share in many projects or products and is the actual unique selling point against competition. Not even the suppliers themselves strive for measurability: Software cost controlling – and thus also risk management – leave a lot to be desired, although there are several useful approaches, such as ROM consumption on hardware that provides for precise code assessment. Complaining about these unfavorable circumstances won’t change things in the foreseeable future.

What will continue to change is complexity, with an upward tendency. The number of developers will not increase proportionally; neither will the available time. Therefore, communication within the team has to be improved. Software has to be more easily changeable and reusable. And finally, it must be possible to locate and solve problems at an early stage.

So when, if not now, should the next change of paradigm in software development take place? Companies should make a change now, as long as they can still somehow master the complexity of their projects. They should learn to swim as long the water is still shallow enough for them to stand on their own feet. Otherwise, they will have to give it a try and find out if they can fight to keep their heads above water of if they simply drown. ♦



*Software complexity increases exponentially with processor performance. 8.000 LOC is the average software size in 8 bit projects. In 16 bit projects, it’s already 80.000 code lines, and one million in case of 32 bit. About every seven years (+/-2), the number of LOCs increases by approximately factor 10. In general: projects with more than 50.000 LOC have to be profitable even in subsequent projects. Only companies that reuse as much code as possible can handle shorter project times and tighter cost frames.*

*Source: VOC*

## Pathfinders for UML

*UML itself does not mean a quantum jump in embedded software. It is simply just a standard notation; however, a very sophisticated one that is accepted worldwide. The successful use of UML depends on a variety of factors, and all of them need to be considered in equal measure.*

Let's go back one decade. At that time, it was structured procedural programming that revolutionized software development and not – contrary to the general opinion – the change from Assembler to C. So it is mainly a change of methods that paves the way to innovation. In case of UML, object oriented programming (OOP) is the basis.

### Powerful Method

One weakness of modern embedded systems is the missing architecture. The consequences are fatal, because the dependencies of functionalities on the time or priority level can no longer be mastered in this way. Code alone cannot visualize the architecture – it needs to be displayed graphically. What serves this purpose better than an established standard like UML?

UML stands for the method of OOP, however, with limitations from an embedded point of view, because OOP origins from the IT world. However, by now the complexity of embedded systems has increased drastically, even in the 16 bit range, and it takes object oriented methods to master it.

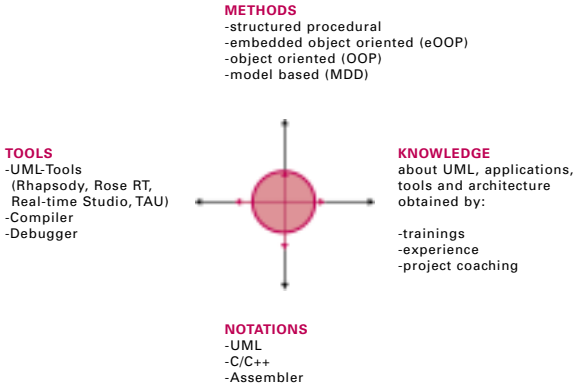
Consequently, we suggest an OOP variant – embedded OOP (eOOP). It is mainly focused on encapsulation and approaches this subject from a completely different angle than IT. Let us show you the difference with two examples:

- A typical IT case: A sales team member needs the telephone number of a customer and starts a database query from the user layer of his PC, probably with a method. This communication mechanism, synchronous from a time-related point of view, does not affect the system, as the sequences are directed from the slow reaction time of the user towards the fast reaction time of the hard disk drive.
- A different embedded world approach: For example, a real-time oriented system gets visualization data via the CAN bus. Let's assume that a CAN routine synchronically uses a method call to get a service from the visualization routine on the interrupt layer. This would provide for a coupling of the time behavior of the visualization layer (ms) with the time behavior of the CAN routing ( $\mu$ s). The result would be a complete paralysis of the interrupt layer.

These examples clearly show the importance of encapsulation with eOOP in the embedded market.

## Efficient Tools

Those who dare to change from structured procedural programming to eOOP and thus from C to UML have to consider many other aspects, including tools that extensively support the method and notation.



*Method, notation, tools and knowledge have to be involved in equal measure in order to get a real quantum jump in software development. The more distinct each of these aspects are, the more successful the UML project. The least distinct aspect is the bottleneck. Source: Willert Software Tools 2004*

An experienced surgeon won't operate successfully with only a blunt scalpel at hand. In other words: methods often leave much scope and a developer could theoretically also do object oriented programming in Assembler. The question is: how efficient would he be?

## Precious Knowledge

Eventually, the potentials of method, notation and tools will end up in smoke if there is a lack of suitable know-how within a company. This knowledge can be built up and extended step by step, e.g. with focused training and external project coaches. Today, there is backlog demand especially for the knowledge of architecture which is the basis for eOOP and thus for UML.

### Clearly differentiate between terms

Imagine an everyday communication situation:

**Notation:** We could not communicate without a common language, such as German or English.

**Method:** People communicate in different ways: in writing or personal.

**Tools:** Understanding is impossible without mouth and vocal chords.

**Know-how:** If you don't have a sufficient understanding of the grammar of a language, you will have a hard time making yourself understood.

## Strategy of Small Steps

*First of all, it takes cause studies if you're seeking new impulses for your own software development. For example, software depends on processes, methods, tools, notations – and the overall architecture. And exactly the latter is in disorder today. However, UML can already help you solve these problems.*

In modern systems, complexity can only be mastered from a bird's eye view. On a high level of abstraction, structures have to be generated that can be divided into clearly decoupled system components.

### From Diagram to Code

Developers often reach their limitations when implementing a behavior described with UML. Their colorful diagrams sadly hang on the office walls while the team has already returned to C and thus many times to structure oriented programming. Again, the well known breach between design and documentation on the one hand, and code on the other. The main reason is the architectural design. In many cases, the runtime behavior designed with UML cannot be transferred successfully to an implementation equivalent due to lack of experience. Developers have to face these challenges:

- They require a comprehensive architectural design to precisely transfer behavior.
- UML can represent more than C does, e.g. also the design of a runtime behavior. In order to fully exploit the advantages of rapid changeability, it has to be implemented flexibly and consistently in C or

C++. First of all, this requires a good knowledge of UML, as well as a suitable runtime system and a predefinition on how to convert the UML constructs.

- Especially in their first projects, developers are confronted with experience and knowledge gaps. They often don't fully understand the notation UML. Elements that influence the architectural design are often not implemented in a suitable way.
- There is often a lack of eOOP experience.
- Those who only work on the UML level do not get information on design characteristics. For example, how can runtime or size be transferred to C or C++? If there is no such feedback, the development results obtained are often unrealistic.

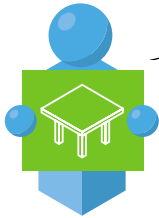
The C Programmer



*Now we are going to build this round table*

Implementations on C basis require additional information, e.g. on system architecture and runtime, which cannot be fully expressed in C syntax.

The UML User (inexperienced)



*Now we are going to build this round table*

The descriptive power of UML exceeds that of C. However, without experience or feedback, the use of UML often leads to redundant or misleading designs and thus to a breach between design and implementation.

The UML User (experienced)



*Now we are going to build this table*

One basic requirement for return on investment is the consistency between design and implementation. That takes experience or permanent feedback on consistency.

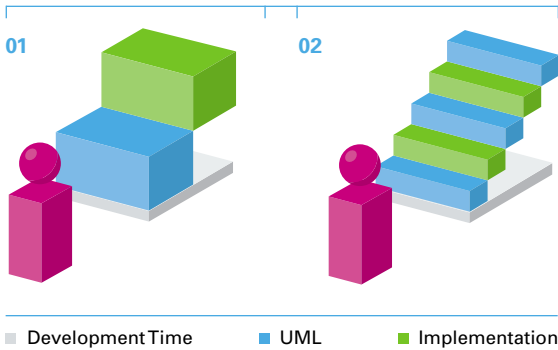
When companies reach the point where they feel like throwing in the towel, they should remind themselves that they can obtain UML experience step by step without seriously putting their projects at risk. However, an introduction that is focused only on documentation is not recommended. Automatic code generators, for example, provide clear feedback. Those who stop when they encounter the boundary to implementation will never know if they have worked with UML the right way.

As the functionality of embedded solutions is very much a function of physical quantities, like runtimes or memory sizes, the relationship between model and reality has to be established as quickly as possible. It is not really useful to clumsily design a pompous castle to find out later that the building ground is too small and the available material does not fit.

Therefore, the team should develop with eOOP in a lean and consistent way. To this end, they pick a small segment from the project and develop it down to production code generation. In many cases, this requires only one or two development places, and every company has employees that are keen on capturing new territory.

Via a suitable UML tool, these pioneers will soon get feedback on whether the right code can be generated from the design. They can share their experience with the team and handle even more segments in a subsequent project. In this way, the complete object oriented design can be transferred quickly with the help of UML. So how do you eat an elephant? Simply bite by bite - break it down!

Source: Willert Software Tools 2004



**01 Full Risk:** *Developers who transfer the complete analysis and design using UML but then do conventional implementation risk a breach between design and code.*

**02 Optimum:** *UML is introduced best with clearly differentiated sub-functionalities that are implemented completely. Afterwards, the company can already cover several project segments and developer work places and thus introduce UML stepwise without taking any risk. Source: MicroConsult 2004*

## Divide and Conquer

Even Julius Cesar got his huge empire under control using the simple trick “divide and conquer”. The structured procedural method with C will not help here, because, unlike UML, it cannot really decouple the software components. For example, it often considers only one aspect of real-time embedded systems, whereas UML presents and describes the four main dependencies:

- behavior
- time
- data flow
- priority

In procedural programming, there are further inter-component connections “below the surface”. A change on one component can result in serious runtime changes on another. Thus, the “divide and conquer” principle would become invalid. With object based programming and UML, however, it is only a step towards encapsulation in runtime oriented embedded projects: The interfaces of the individual components are related to only one aspect, e.g. data flow. Behavior, time and priority are not affected.

The transition from procedural to object based programming will thus solve most of today’s problems and has a huge potential for innovation that is urgently need in software development. The foundation stone of this change will not be C++, as often assumed, but rather the runtime design. Thus, the runtime system (Real-Time Operating System, RTOS) will play

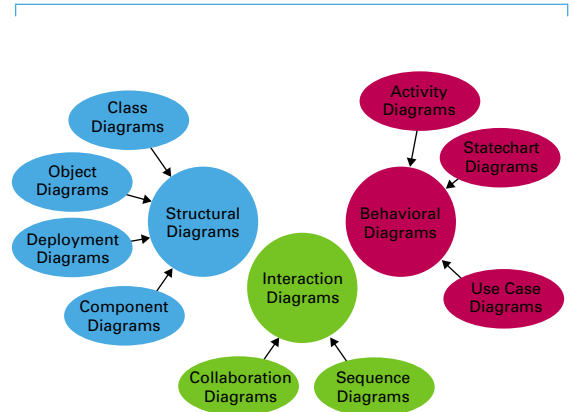
a leading role.

These advantages can even be exponentiated with the simultaneous use of an object oriented notation, such as C++ or UML. There are more and more developers who comprehend this, and this is reflected by the rapidly growing demand for object oriented methods and tools over the past few years.

So the question is no longer: Is OOP helpful for embedded developers, and will it be accepted in future? But rather: How much longer can we afford to do without object based programming with an object based runtime architecture, and when will it become a real competitive factor? Or has it already become one?

### A Picture is Worth A Thousand Words

Last but not least, UML offers the advantage of visual modeling. Using a line oriented editor, it is hardly possible to overview all coherences of an application, such as time and functional behavior or data flow, especially by the customer, service provider or management. With UML, the brachiating from line to line, from module to module, is definitely a thing of the past. UML graphically displays complex coherences which is an advantage also for the team, because everyone can thus broaden their horizon and understand the major coherences. ♦



Visual modeling with UML: All people involved, even those without technical experience, can understand what the project is mainly about.

*Source: Willert Software Tools 2004*

## Economical of ROM and RAM

*Where memory and computer performance play a subordinate role, UML has certainly become first choice by now. However, development companies are afraid of using it for applications with limited resources. For example, a project with a 16 bit micro-controller has to get by with a minimum of 64 Kbyte ROM and one Kbyte RAM. If the code generator overhead is too large for small or medium-size systems, there will be a breach between design and implementation. Willert Software Tools' consulting projects have demonstrated that UML can score even in 16 bit applications.*

About three years ago, 16 bit users acquired a taste for it: At that time, UML facilitated documentation and communication. Projects could be conveyed more concisely and clearly with diagrams, both within the company and to the customer. First experience made them ask for more: They wanted to test system consistency and logical correctness by means of model simulation, to be able to identify operating errors at an early stage. Consequently, the next step should be to generate code for the target system from the model. These hopes were not always fulfilled – for many different reasons.

First of all, UML is an exact notation and requires precise procedures. If you have obtained the required knowledge from books or theoretical training, this might be a first introduction; however, getting into practical use will cause problems. Unless you get feedback in this first phase, methodical errors will accumulate and exponentiate.

Common problem sources are e.g. asynchronous or

synchronous communication in sequence charts or a missing default state in nested states. Project members could sometimes generate a design and even simulate part of the model. But then they continued with conventional programming, i.e. manually, in C. This led to an irrevocable loss of an important UML approach – object oriented design. The UML design and the code realized in C drifted apart. The result was compulsive and momentous: Application quality suffered seriously and many projects had to be restarted completely.

Not so long ago, tool suppliers provided only little support, especially for users of 16 bit microcontrollers working with e.g. models from the Infineon C167 or Mitsubishi M16C family. The first coding attempt for a UML project by Willert Software Tools failed after three months: Despite wholehearted promises by the tool supplier, the generated code failed to work with the 16 bit C compiler.

In a second try with a different tool, the supplier promised adaptation within a few days. After four weeks, a UML design could run on the C167 hardware: a simple "Embedded Hello World" (LED blinking). It required 56 Kbyte ROM and one Kbyte RAM. Some further classes and state diagrams later, the sonic wall of 100 Kbyte ROM was penetrated. Due to the 256 Kbyte ROM limit, this solution was not practical for the customer. After four more months with uncountable night work, a pilot project could finally be started.

The adaptation of a UML framework as a basis for code generation in 16 bit systems consequently turned out to be much more complex than expected,

because nearly all UML tools were designed for the 32 bit world. Today, these tools are much more flexible and can be used for projects of various sizes.

### Lean Code

Overhead by code generation should always remain below 50 Kbyte. In the Willert project, it was far below 30 Kbyte ROM, i.e. approximately 100 byte RAM, including RTOS. Approximately 10 to 30 percent of additional code has to be calculated on top due to object oriented programming structures. In larger projects, it is only 0 to 20 percent, because the structure is maintained to a higher extent. However, in general these values strongly depend on the work style of the developer, i.e. his experience. In our example, the object oriented system was based on preemptive multitasking and thus performed its task without any runtime losses. Disadvantage: The speed advantage had to be bought with several hundred bytes of RAM. By now, however, there are many runtime system alternatives providing for a 4 to 20 Kbyte overhead only.

Layer		Code increase	Reaction times
System Layer		Overhead Code 10 to 30% at project start    0 to 20% in long-term projects	100 µs
RTS \ RTOS layer and framework OSAL		Footprint	4 to 20 KByte
Interrupt layer	Driver layer	no increase	1 µs
Hardware layer			

**Platform:** Infineon C167, 20 MHz clock frequency, 16 bit external memory

### Footprint and Reaction Times with UML

One vital decision factor for using UML is the related overhead. We have done some measurements for you: The widely used Infineon 16 bit CPU was the hardware platform. "Rhapsody in C" was used as UML tool with code generation, and the operating system CMX RTX was the runtime system. The reaction times between setting a bit on the hardware and resetting the bit after the expected reaction across the individual layers were measured. The architectural design that was encapsulated best was realized with the available methods.

*Source: Willert Software Tools 2004*

### Know the Limits

UML is powerful, but it has got its limits. For example, the interrupt service routines should continue to be programmed conventionally. The developer can then define the priorities for the most important routines on top of the RTOS and the UML framework. In this way, interrupt latency is not affected, and full performance is maintained. This means, however, that the programmer has to do without the convenience of the RTOS and object oriented features within the routines.

Not all projects are suitable for the use of code generators, as in the above mentioned example shows. Single chip solutions with less than 4 Kbyte RAM are problematic. But even here, it is still possible to design systems based on UML and code generation by specific abdication of certain UML constructs. However, these projects are not characteristic for beginners and require a lot of construct handling experience.

In general, UML with automatic code generation for C, combined with an RTOS, is about to become the ideal method for getting started with object oriented programming even for smaller systems.

### Tool Integration

Many developers use modeling tools for time-continuous systems to implement algorithms, e.g. for controllers. They can be integrated with UML tools. The problems of tool coordination, simulation synchronization and distributed data management are solved with EXITE from EXTESSY. The platform provides for the co-simulation and distributed simulation of any models in PC clusters of any size. For the time being, EXITE offers interfaces to MathWorks Simulink, MATLAB Real-Time Workshop Executables, ARTiSAN Real-Time Studio, I-Logix Rhapsody, Synopsis Saber, ETAS ASCET and Dynasim Dymola. Moreover, the platform features a C/C++ API for any simulation. In this way, different CASE tools for different tasks can be implemented in a complete and consistent process.

## UML Real-Time CASE Tools – Overview

### Performance of all Tools

- UML 1.x to 2.0
- Code generation (C, C++, Java and Ada) from state, activity and class diagrams
- Object based animation, simulation and debugging on the development system or directly on the target hardware
- Automatic generation of documentation
- Configuration management

### Additional Performance by Certain Tools

- Team-based development
- Test support
- Implementation of (real-time) operating system, (cross) compiler and debugger

### Suppliers – an Overview

<b>ARTISAN</b> Real-time Studio	<a href="http://www.artisansw.com">www.artisansw.com</a>
<b>I-Logix</b> Rhapsody	<a href="http://www.ilogix.com">www.ilogix.com</a>
<b>IBM Rational</b>	<a href="http://www.rational.com">www.rational.com</a>
Rational Rose RealTime	
<b>Telelogic</b> TAU	<a href="http://www.telelogic.com">www.telelogic.com</a>
<b>Willert Software Tools</b>	<a href="http://www.willert.de">www.willert.de</a>
Embedded UML Studio 16bit (based on Rhapsody)	

## UML 2.0 for Embedded Software

*In principle, UML 1.x can be used for embedded systems and has provided all the advantages described in this Trend Guide for many years. However, as always, there are requests and ways for improvement – and that's why UML 2.0 was developed. The following sections describe the novelties that are relevant especially to embedded systems.*

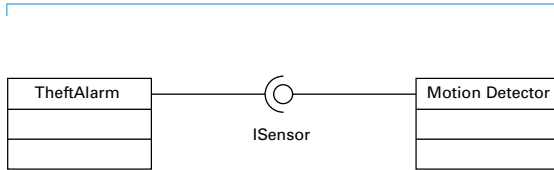
### Hierarchical Decomposition

UML 2.0 provides for the elegant linking and navigating between different abstraction levels within a system. The following features were extended or added:

- structured classes (structure diagram)
- ports with required and provided interfaces
- links for ports
- lifeline decomposition in sequence diagrams
- referencing in activity and sequence diagrams

### Interfaces

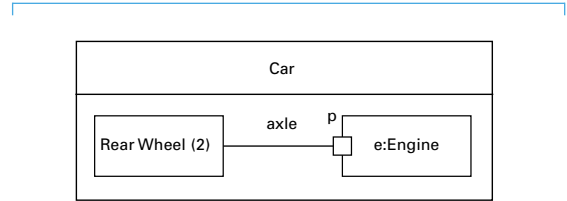
UML 1.x already works with interfaces. UML 2.0 additionally introduces Required and Provided Interface which are used to define ports.



*TheftAlarm* enables the activation of a sensor. At the time the class *TheftAlarm* is developed, it is not yet clear which sensors can be activated. The so-called Required Interface *ISensor* states that a following sensor has to provide an *ISensor* Interface. This Required Interface is represented by a so-called socket. The *Motion Detector* implements the *ISensor* Interface. Thus, the *Motion Detector* sees the *ISensor* only as a *Provided Interface*. Source: *MicroConsult, 2004*

**Structure Diagram**

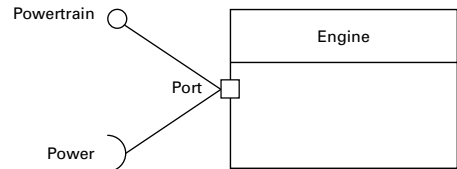
In UML 1.x, the structure view of a project is limited to classes. Object view becomes an issue only with behavioral analysis. Disadvantage: With this method, it is not clear how objects interact in the project. With Structured Class, ports were introduced in UML 2.0. Ports describe the places of interaction between objects. This does not yet define the concrete interface, but only the will to communicate and the intended direction of communication.



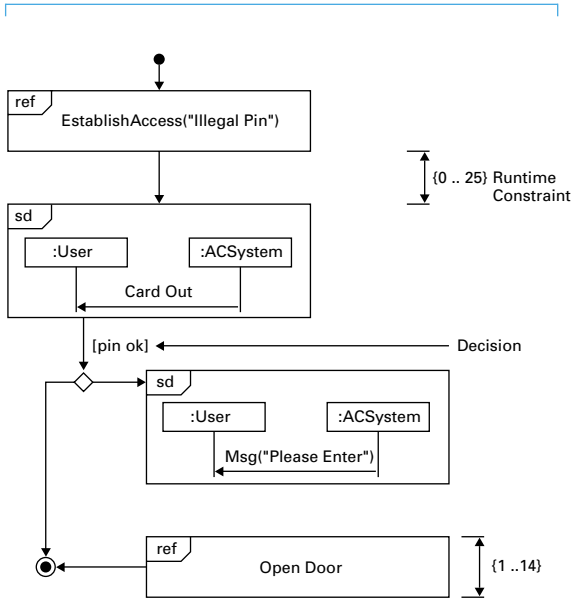
*Example Automobile: This diagram shows the situation "drive to wheels": The engine provides an interface which is not yet defined precisely but guarantees in any case that the wheels are driven.*

Source: *MicroConsult, 2004*

**Ports with Provided and Required Interfaces**



*The engine received a port before. Now, communication is substantiated by the "Powertrain" and "Power" interfaces. Methods with concrete signatures hide behind these interfaces.* Source: *MicroConsult, 2004*



These new elements of UML 2.0 provide for a virtual construction kit. Via plug-and-play, predefined objects are arranged in a Structured Class, and links “wire” their ports with each other and the outside world. The Structured Class can define a subsystem, for example. On the next higher hierarchical level, these objects (i.e. instances of the relevant Structured Class) are combined to an overall system. Developers can use the Structured Class both in the project start phase where classes are not yet concrete, and for design which generates concrete objects from the classes found. ♦

*Example: Activity Diagram for a Door Lock: This diagram lies within a frame. The individual activities for access control, card output, “Please Enter” message and door opening are represented by subordinate frames. A frame can represent a sequence, collaboration, state or activity diagram.*

*Source: MicroConsult 2004*

## 10 Practical Tips

### 01 Take Part in Trainings

Get started with UML by intensive training courses or workshops which, however, need to be tailored to embedded software. 90 percent of all UML training courses are not suitable for embedded developers and thus a misinvestment.

### 02 No half-hearted Start

The risk is your enemy. That's why introduction should commence with small steps as far as your project is concerned. But it should be carried out consistently for one to three work places, with suitable training and the right tools, consistently down to the implementation of the final target hardware. On this experience basis, you can continue to expand UML in your projects and thus keep the risk low.

### 03 Learning from Feedback

You get feedback on your work not only from external project coaches (see item 9) but also by a consistent use of UML down to the implementation stage (fragmentary target implementation). CASE tools with code generation provide a direct feedback on the code generated and thus on how well you're already doing with UML. Remember to run this code also on the target hardware in order to get feedback on the runtime behavior.

### 04 Question Tool Suppliers' Promises

Ask critical questions about the promises made by tool suppliers, for example, regarding fast adaptation of code generation to your system (CPU, com-

piler, RTOS). Let them show you an executable solution and ask for reliable statements on the footprint of the generated code. Unless there is concrete information regarding your target platform, calculate several months of evaluation effort.

### 05 Know the Limits of UML

UML is not an "all-in-one" solution suitable for every purpose. For example, it cannot image any continuous application elements. In this case, use controller models, imaged with MathWorks MATLAB, ETAS ASCET-SD or National Instruments MATRIXx. Expect also some additional work on the generated C or C++ code. Moreover, not all UML constructs are suitable when systems have to economize with resources. The generated code will show you best how a construct influences runtime or code size.

### 06 Avoid Breaches between Analysis, Design and Implementation

These breaches are the most common ROI killers. The use of UML is successful only if production code is generated in combination with reverse engineering (implementation of available C sources) and roundtrip engineering (changes on C level that are automatically taken over in the model). If UML design and code are identical, the system becomes understandable and the components can be reused.

**07 Reuse old Code**

Old code can be reused with a UML model. This is done on the C interface level. Don't try to transfer code to model 1:1 or to convert an object-oriented approach to procedural code belatedly. Just concentrate on the interfaces to the old code to be transferred to the model.

**08 Use a Runtime System**

The architectural design reflects the task behavior of the system. Make sure that the modeled architecture is identical to the behavior on the hardware. This is achieved only by using an embedded operating system or a runtime system with similar characteristics.

**09 Reduce the Introduction Phase with Coaching**

Project assistance by coaching provides for a smooth transition from training to practical use, within a few days or weeks. Coaches will accompany your first project steps and make sure that you're taking the right direction.

**10 Remember Testing**

A good CASE tool assists you during all phases of the project. You can display requirements, carry out analysis and design and of course generate code. Complete your UML model with test cases and generate the required code from the model. This will help you achieve higher test coverage from the start. ♦

**Reduce Costs and Increase Efficiency**

A cost/benefit calculation on UML is difficult to carry out, of course. There are just too many different projects and basic conditions. We still dare to give it a try using a concrete example:

A company has a team of five developers and requires further capacities. It can either

- hire another employee or
- introduce UML

Current customer statements confirm that in the first UML project efficiency increased by between 20 and 30 percent. To be on the safe side, we estimate 20 percent, thus covering the efficiency of an additional employee. Take a look at the costs:

- One additional developer:  
Salary (including additional expenses, excluding vocational adjustment): approx. 80.000 Euro/year  
Consequential costs: approx. 80.000 Euro/year
- Introduction of UML  
5 work places (including training, excluding vocational adjustment): approx. 70.000 Euro  
Consequential costs: approx. 15.000 Euro/year

Consequently, the introduction of UML helps this exemplary company to reduce costs by 65.000 Euro per year. These savings are even more obvious in case of larger teams. Conclusion: In Germany, the high-wage country, it is more cost effective to make the existing team more efficient by using UML. ♦

## UML Step by Step

If you want to succeed in establishing UML step by step, you can get tangible support during all stages of the establishment process.

### **Orientation: What the hell is UML?**

UML is more or less a catchword to you. Now you want to know more precisely what's behind it. Take time for a day and visit the Roadshow by MicroConsult and Willert Software Tools. For current dates and locations, please visit [www.microconsult.de/events\\_press](http://www.microconsult.de/events_press)

### **Information: The team wants to know!**

Probably not all developers and decision-makers in your company can take part in a several days' UML training. But you and your colleagues have questions you're eager to get answered? We and our partner, Willert Software Tools, will be happy to come to see you for half a day at no cost, to answer your questions.

### **Training: I want to know!**

You already have some information on UML, but cannot get a clear picture of how you can benefit in concrete projects? Or you want to extend your know-how in practice? Join us for an intensive practical training course: [www.microconsult.de/trainings](http://www.microconsult.de/trainings)

### **Project coaching: Here you go!**

You have decided to use UML and want to start your first project? Or you have already tried several times and are not satisfied with the results? Get help to help yourself. Contact our project coaches for new solution processes and support when putting your knowledge into practice.

### **More project coaching: You can do even better!**

You're already using UML, e.g. for documentation or design? And you want to use it more? This is the right time for advanced training, reviews or coaching.

Your contact for project coaching, training and consulting:

Peter Siwon, MicroConsult  
Tel.: +49 (0)89 / 45 06 17-44  
[p.siwon@microconsult.de](mailto:p.siwon@microconsult.de)

Klaus Peter Rosenthal  
Tel.: +49 (0)89 / 45 06 17-62  
[k.rosenthal@microconsult.de](mailto:k.rosenthal@microconsult.de)

## Book Tips

**Doing Hard Time**, Developing Real-Time Systems with UML, Objects, Frameworks and Patterns by Bruce Powel Douglass. With CD-ROM. Addison Wesley 1999, 800 pages, ISBN 0-20149-837-5.

*Douglass describes a programming method that is successful especially in case of real-time embedded systems. Application developers learn how to use common techniques of object oriented software programming. Douglass covers subjects like real-time frameworks or behavior patterns in connection with UML.*

**Real-Time UML**, Developing Efficient Objects for Embedded Systems by Bruce Powel Douglass. Addison Wesley 2004, 3. edition, 694 pages, ISBN 0-32116-076-2.

*The current edition of this book explains UML 2.0 in easily understandable English and keeps the balance between topical depth and relaxing sections. It covers, amongst others, requirements analysis, definition of object structures and behavior as well as mechanistic and architectural design.*

**Agile Softwareentwicklung für Embedded Real-Time Systems mit der UML (Agile Software Development for Embedded Real-Time Systems with UML)** by Peter Hruschka and Chris Rupp. Hanser 2002, 208 pages, ISBN 3-44621-997-8.

*The book propagates a shift of the development cycle from a slow but very accurate over-precision and modeling down to the smallest detail towards a software development strategy that is sufficiently precise but can be dynamically adapted to market requirements.*

**UML 2 glasklar (Crystal clear UML 2)** by Mario Jeckle, a.o., Hanser 2003, 436 pages, ISBN 3-44622-575-7.

*The authors provide an extensive overview of all 13 diagram types and the UML elements as well as the right relation to an efficient practical use of UML 2 for own projects.*

**UML for Real**, Design of Embedded Real-Time Systems by Luciano Lavagno, a.o., Kluwer 2003, 369 pages, ISBN 1-40207-501-4.

*Lavagno covers today's possibilities and limitations of UML as a medium for specifying and implementing real-time systems. At the same time, he shows future options for the notation. Select application examples round off his work.*

**Die UML 2.0**, Kurzreferenz für die Praxis (UML 2.0 Short Reference for Practical Use) by Bernd Oesterreich. Oldenbourg 2002, ISBN 3-48627-213-6.

*A reference book tightly focused on the elements that are important for daily modeling tasks. The individual UML elements are classified along definition, description, notation and example.*

**Leitfaden für kreative Softwareentwicklung (Guideline for Creative Software Development)**

by Andreas Willert. BoD 2001, ISBN 3-8311-0594-4.  
*Set book for every developer or project leader in the embedded software development arena. The author highlights software design as the most important prerequisite for optimum project realization and the developer's creativity as key factor for success.*

## Web Tips

### **[www.embedded-forecast.com](http://www.embedded-forecast.com)**

Free download: A new white paper on UML 2.0 from the American consulting and research company Embedded Market Forecasters.

### **[www.gigascale.org/metropolis/EmbeddedUML.whitepaper.v7.External.PDF](http://www.gigascale.org/metropolis/EmbeddedUML.whitepaper.v7.External.PDF)**

Free download: The white paper "Embedded UML: a merger of real-time UML and co-design", with the focus on hardware/software co-design.

### **[www.ilogix.com](http://www.ilogix.com)**

I-Logix website containing extensive, informative white papers, e.g. by the "UML Guru" Bruce Powel Douglass.

### **[www.jeckle.de](http://www.jeckle.de)**

UML expert Prof. Mario Jeckle, member of various standardization committees at OMG and W3C, provides an assessment of the UML 2.0 potential.

### **[www.uml.org](http://www.uml.org)**

The Unified Modeling Language Resource Page of the Object Management Group (OMG) with many helpful links. OMG is an open, non-profit consortium that develops specifications for interoperable business applications, such as UML.

### **[www.umlconference.org](http://www.umlconference.org)**

Website of the 7th UML Conference 2004 in Lissabon from 11. to 15. October.

### **[www.uml-embedded.com](http://www.uml-embedded.com)**

A portal by Berner & Mattner Systemtechnik, IVM Engineering and Willert Software Tools.

### **[www.uml-forum.com](http://www.uml-forum.com)**

Community and knowledge portal with current information on UML: notation changes, exercises, tools, events and other sources.

### **[www.uml-zone.com](http://www.uml-zone.com)**

Sub-site of the DevX.com developer forum.

### **[www.willert.de](http://www.willert.de)**

The Willert Software Tools website, practical information in the "Know-How" section, e.g. on software quality and embedded test.

Further interesting UML information can be found on the websites of leading embedded magazines:

### **[www.cera2.com](http://www.cera2.com)**

### **[www.embedded.com](http://www.embedded.com)**

### **[www.embedded-software.com](http://www.embedded-software.com)**

### **[www.reed-electronics.com/ednmag/](http://www.reed-electronics.com/ednmag/)**

## Related Training Courses

### Object Oriented Analysis with UML

**Participants:** Project leaders in software development, application programmers, software developers in the embedded arena.

**Prerequisite:** Programming experience (e.g. CHILL, Fortran, C, C++)

**Target:** Competent use of analysis and design methods and of the UML display format

**Content:** Elements of object oriented programming; introduction in basic object oriented concepts and display using UML class notation; dynamic behavior of object oriented software; conversion to different programming languages; UML 2.0 innovations.

**Duration:** 4 days

**Price:** 1.480 Euro + VAT

Please refer to [www.microconsult.com](http://www.microconsult.com) for current training dates and the complete training program.

### Object Oriented Analysis for Embedded Systems

**Participants:** Software developers of embedded systems.

**Prerequisite:** Project experience with embedded systems and basic UML knowledge (see training Object Oriented Analysis with UML).

**Target:** Correct assessment of the use of UML for embedded systems; systematic development of software systems – from requirements analysis to design.

**Content:** UML and embedded systems; S.P.E.E.D process model; requirements analysis; analysis; real-time analysis; design; project example as practical training.

**Duration:** 5 days

**Price:** 1.990 Euro + VAT

Please refer to [www.microconsult.com](http://www.microconsult.com) for current training dates and the complete training program.

## Related Training Courses

### Tool Coupling and Co-Simulation in System and Software Development

**Participants:** System and software developers; application developers; project leaders; system integrators.

**Prerequisite:** Basic UML knowledge (see training Object Oriented Analysis with UML), or knowledge of the description of reactive systems; project experience in technical systems development.

**Target:** Combination of the models for individual tasks; transferring the theoretical knowledge of the coupling of UML tools and automatic control technique (block diagrams, signal flow charts) to practical use.

**Content:** Elements of automatic control system modeling; using UML to model technical systems; combination of description languages; simulation tool coupling.

**Duration:** 3 days

**Price:** 1.150 Euro + VAT

Please refer to [www.microconsult.com](http://www.microconsult.com) for current training dates and the complete training program.

### Software Projekt Management

**Participants:** Project leaders; project team members.

**Prerequisite:** Project experience.

**Target:** Knowledge of methods and processes in development projects; use of this knowledge to optimize own projects.








**Content:** Definition and methods of project management; project organization; proven process models in the product development process; realistic project planning; techniques and processes of project controlling; simulation with SimulTrain.

**Duration:** 5 days

**Price:** 1.990 Euro + VAT

Please refer to [www.microconsult.com](http://www.microconsult.com) for current training dates and the complete training program.

## Index of Partners

Focus	Company	Logo	Info	Contact
UML Tools	ARTiSAN Software Tools GmbH Eupener Str. 135 – 137 50933 Köln		<a href="http://www.artisansw.com">www.artisansw.com</a>	Christiane Kapteina Tel.: +49 (0)221 / 48 52 2-60 <a href="mailto:christiane.kapteina@artisansw.com">christiane.kapteina@artisansw.com</a>
Simulation and Development for Technical Systems, UML Consulting	EXTESSY AG InnovationsCampus Major-Hirst-Straße 11c D-38442 Wolfsburg		<a href="http://www.extessy.com">www.extessy.com</a>	Wolfgang George Tel. +49 (0) 5361 / 89 76 025 <a href="mailto:w.george@extessy.com">w.george@extessy.com</a>
UML Tools	I-Logix Deutschland GmbH Bahnhofstr. 39 85591 Vaterstetten		<a href="http://www.ilogix.com">www.ilogix.com</a>	Wolfgang Leimbach Tel. +49 (0)8106 / 37 96 60 <a href="mailto:wleimbach@ilogix.com">wleimbach@ilogix.com</a>
Development tools and environments, Emulators, Debuggers	iSYSTEM AG Carl-Zeiss-Str. 1 D-85247 Schwabhausen		<a href="http://www.isystem.com">www.isystem.com</a>	Erol Simsek +49 (0)8138 / 69 71-50 <a href="mailto:sales@isystem.com">sales@isystem.com</a>
Training, Coaching and Project Assistance	MicroConsult GmbH Rosenheimer Str. 143 b 81671 München		<a href="http://www.microconsult.com">www.microconsult.com</a>	Peter Siwon Tel.: +49 (0)89 / 45 06 17-44 <a href="mailto:ps@microconsult.de">ps@microconsult.de</a>
On top Solutions for System on Silicon Debugging	pls Programmierbare Logik & Systeme GmbH Technologiepark		<a href="http://www.pls-mc.com">www.pls-mc.com</a>	Heiko Riessland Tel.: +49 (0)35722 / 384-0 <a href="mailto:info@pls-mc.com">info@pls-mc.com</a>
UML Solutions for Systems with Limited Resources	Willert Software Tools Herminenstr. 17 B 31675 Bückeburg		<a href="http://www.willert.de">www.willert.de</a>	Andreas Willert Tel.: +49 (0)5722 / 96 78 60 <a href="mailto:awillert@willert.de">awillert@willert.de</a>

**Publisher:**

MicroConsult GmbH  
Rosenheimer Str. 143 b, 81671 Munich  
Tel. 089 / 45 06 17-0, Fax 089 / 45 06 17-18  
Internet: [www.microconsult.de](http://www.microconsult.de)

**Object Management:**

Sabine Häring, MicroConsult

**Authors:**

Martin Stockl, I-Logix  
Klaus-Peter Rosenthal, MicroConsult  
Andreas Willert, Willert Software Tools

**Editorial Management:**

Eva Schulz, actimedia

**Art Direction:**

Florian Gmach, entity38

**Handling:**

entity38



# Vom UML-Modell bis zum Target-Code und zurück

- UML2.0 Entwicklungs-  
umgebung für Echtzeit-  
Systeme
- Reverse Engineering  
Debuggen auf UML-  
und Code-Ebene
- Embedded-Targeting
- Team Collaboration
- optimal skalierbar
- praxiserprobt

## I-Logix

I-Logix Deutschland GmbH  
Bahnhofstrasse 39  
D-85591 Vaterstetten  
(bei München)

Telefon 08106-379-660  
Telefax 08106-379-666

Internet: [www.ilogix.com](http://www.ilogix.com)  
mailto:[sales@ilogix.de](mailto:sales@ilogix.de)

**Trend Guide Media Partner:**

**ELEKTRONIK  
PRAXIS**

**[www.elektronikpraxis.de](http://www.elektronikpraxis.de)**



**MICRO CONSULT**

MicroConsult GmbH • Schule für MicroElektronik & Informationstechnologie  
Rosenheimer Straße 143b • D-81671 München • Tel.: (089) 45 06 17-71  
Fax: (089) 45 06 17-17 • [www.microconsult.de](http://www.microconsult.de)