Software Development for Embedded Systems

J. Tick
Óbuda University / John von Neumann Faculty of Informatics / Software Engineering Department, Budapest, Hungary
tick@uni-obuda.hu

Abstract— **According to some estimates 98% of the presently operating computer systems all over the world are embedded systems. These paper examines the convergence of the PC-friendly software development and the state of art of embedded system-software development in all three ranges traditional low-performance, middle-performance and new type, high performance embedded systems. In the last part the paper deals with the UML-RT – the new way to software development for high performance embedded systems.**

I. Introduction

Embedded systems are small-sized, microcomputer-based independent systems which manage various equipment. [3] These pieces of equipment range on a very wide scale and have an important role in our life. We meet them in cellular phones, microwave ovens, automotives, video recorders, watches, games and in several other devices. [2]

The history of embedded systems has been ever evolving and shows a faster pace than the development of personal computers lately. It is shocking that in the latest years approximately 100 million microprocessors were sold annually, which were built into desktop computers, mainly PCs. At the same time 3 billion such microprocessors have been sold that were incorporated into embedded systems.

This huge number implies that the development of software products for embedded systems has become a serious and good business. While well-developed methodologies, tools and software directories are available for software development for large systems, the software developers face the same problems when developing for embedded systems as they have already overcome during the evolution of large systems. Embedded systems are the most world wide spread computer applications.

According to some estimates 98% of the presently operating computer systems all over the world are embedded systems [1]. The structure, the application fields and the developing of the hardware and software components of these systems very much differ from the world of PCs. In the following paragraphs we will examine such features of embedded systems that become constraints in the evolution of the development methodologies of software products indispensible for such systems.

II. Convergence of the PC-firendly software development

The software development of non-embedded systems differs to a large extent from the development methodologies used in case of embedded systems. In order to decrease the performance-deviations the older middle-category computers and the PC-category applications out of the presently used applications are regarded as non-embedded systems. Especially in case of the latter it can be argued that software development are highly backed not only by applied languages but by methodology and by tools (CASE-Tool) as well. Even if we do not go back to the world of structured methodologies only to the proliferation of object-oriented paradigm, it can be noticed that (fig 1. and fig 2.) a wide range of applied languages and methods were available showing a definite and strong convergence.

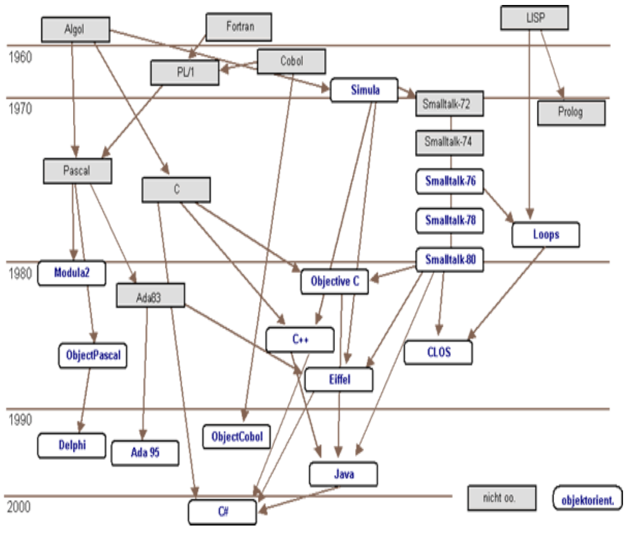


Figure 1 Development of programming languages
source: www.oose.de

The colorful spectrum of languages has disappeared and practically two, very similar programming languages have taken the lead. Most of the application developments are conducted in JAVA and C#.

Considering the methodology side, Rebeca Wirrfs-Brock at all Responsibility Driven Software Engineering, Jacobson: Object-oriented Software Engineering, Raumbought at all: Object Modelling Technique are the outstanding ones out of the earlier methodologies. Via these more significant methodologies the Booch-Jacobson Raumbought (three amigos)-labelled UML (Unified Modell Language) has emerged, which by now has already become an entirely accepted developing approach and notation system. Naturally RUP (Rational Unified Process) - as the methodology - is paired with this modeling language.

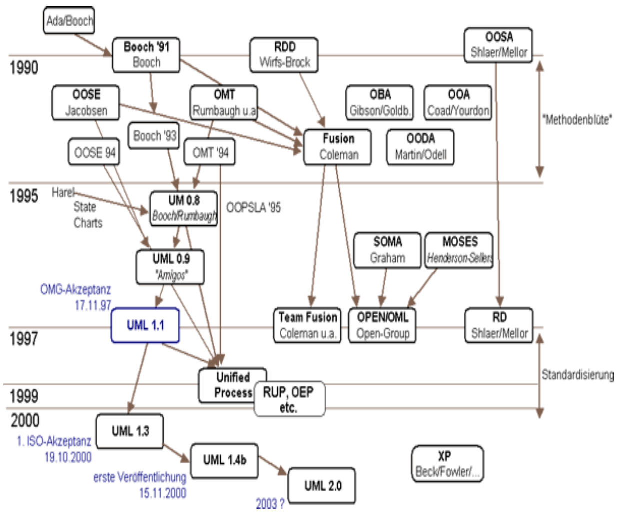


Figure 2 The evolution of software development methodologies source:www.oose.de

This overwhelming success can be thanked to the UML's quite rich support in tools (Fig 3.) and, of course, its extremely strong expressive power. The fact that vast of models-submodels have been developed and piled up lately, cannot be neglected, which quite effectively support Software Reuse.

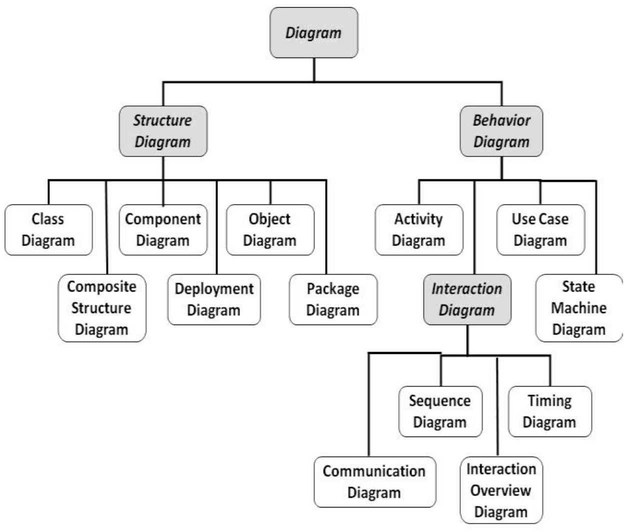


Figure 3 The structure of UML

III. Software Development for Embedded Systems

As we saw in PC environment the developing methodologies converge into one point. However, in case of embedded systems they converge into two points. The features of the traditional embedded systems and the new systems are described in the following.

A. *Typical software develoment method of traditional low-performance embedded systems*

The low-category layer of these systems are the very cheap, low performance, very simply structured systems in mass production, most of which entails a one-chipped microcomputer. The processor itself is typically of 8 bits, its clock frequency ranges between 0.1 MHz and 4-8 MHz. Its storage capacity is from some hundred bites up to 2 Kbyte, the managed I/O ports are usually digital, between 4 and 8. Microchip 16f84 is one of the examples to be mentioned, which is widely used in these systems.

The software is developed at assembly level or with the help of a simple C compiler. The comfortable developing on foreign platform (usually PC) is typical. The programme can be loaded to the microcomputer through some simple serial interface and/or electronically or can be burned with masking technique.

The phases of development are not specially supported by methodologies, the available developing tools are company specific ones. Its basic components are the editor, the assembler, the simple debugger, the support library and the downloader (Fig 4.). The course of application and the development are typically simple; the realized algorithms are not complicated.

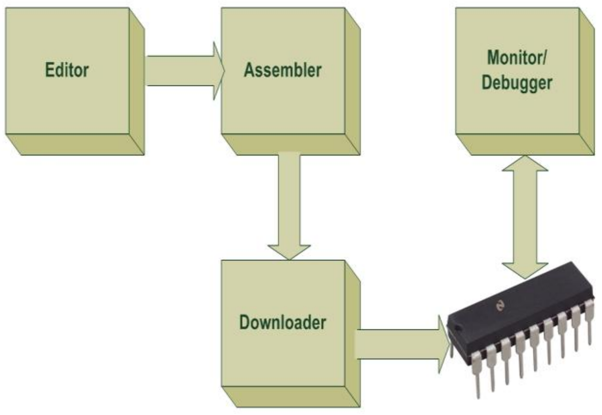
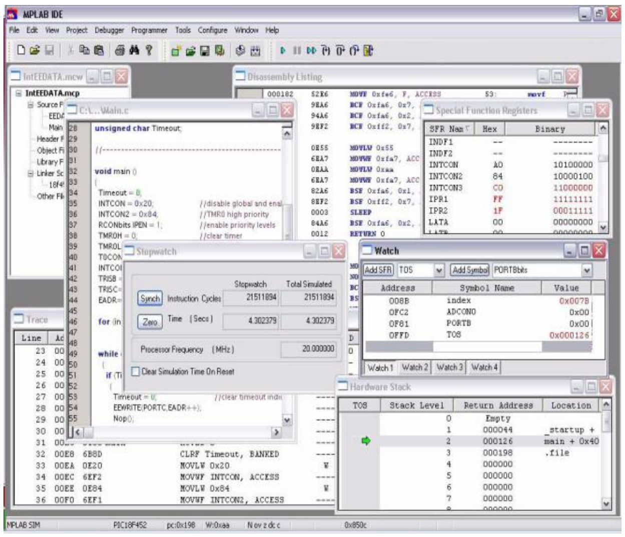


Figure 4 Scheme of the typical software development tool of a low-performance embedded system

B. *Typical software developmentmethod of traditional middle-performance embedded systems*

These systems have favorable price/performance attribute, their central component is a quite complicated, on one-chipped microcomputer.

The processor itself is typically of 16 bits, with RISC architecture, its clock frequency ranges between 4-8 MHz and 40 MHz. Its storage capacity is between 2 and 8 Kbyte, the managed I/O ports are analogue and/or digital, and their number goes from 8 up to 32. Several other inboard services like multi-layer interrupt possibilities, various timers, counters, fast analogue-digital converters to ensure distinct communication possibilities are added.



```
unsigned char Timeout;

//---------------------------------------------

void main ()
{
 Timeout = 0;
 INTCON = 0x20;           //disable global and enal
 INTCON2 = 0x84;          //TMR0 high priority
 RCONbits.IPEN = 1;       //enable priority levels
 TMR0H = 0;               //clear timer
 TMR0L
 T0CON
 INTCO
 TRISB =
 TRISC=
 EADR=

 for (in

 while
 {
  if (Ti
  {
   Timeout = 0;           //clear timeout indi
   EEWRITE(PORTC,EADR++);
   Nop();
```

```
000182  52E6   MOVF 0xfe6, F, ACCESS       53:   movf
        9EA6   BCF 0xfa6, 0x7,
        94A6   BCF 0xfa6, 0x2,
        9EF2   BCF 0xff2, 0x7,

        0E55   MOVLW 0x55
        6EA7   MOVWF 0xfa7, ACC
        0EAA   MOVLW 0xaa
        6EA7   MOVWF 0xfa7, ACC
        82A6   BSF 0xfa6, 0x1,
        8EF2   BSF 0xff2, 0x7,
        0003   SLEEP
        84A6   BSF 0xfa6, 0x2,
        0012   RETURN 0
```

SFR Name	Hex	Binary
INDF1	--	--------
INDF2	--	--------
INTCON	A0	10100000
INTCON2	84	10000100
INTCON3	C0	11000000
IPR1	FF	11111111
IPR2	1F	00011111
LATA	00	00000000

	Stopwatch	Total Simulated
Instruction Cycles	21511894	21511894
Time (Secs)	4.302379	4.302379
Processor Frequency (MHz)		20.000000

☐ Clear Simulation Time On Reset

TOS PORTBbits

Address	Symbol Name	Value
008B	index	0x007B
0FC2	ADCON0	0x00
0F81	PORTB	0x00
0FFD	TOS	0x000126

Line	Ad			
23	00			
24	00			
25	00			
26	00			
27	00			----
28	00			----
29	00			----
30	00			----
31	00			----
32	00E8	6B8D	CLRF Timeout, BANKED	----
33	00EA	0E20	MOVLW 0x20	W
34	00EC	6EF2	MOVWF INTCON, ACCESS	----
35	00EE	0E84	MOVLW 0x84	W
36	00F0	6EF1	MOVWF INTCON2, ACCESS	----

TOS	Stack Level	Return Address	Location
	0	Empty	
	1	000044	_startup +
➡	2	000126	main + 0x40
	3	000198	.file
	4	000000	
	5	000000	
	6	000000	
	7	000000	

MPLAB SIM PIC18F452 pc:0x198 W:0xaa N ov z dc c 0x850c

Figure 5. User interface of the Integrated Development Environment (Microchip) source: www.microchip.com

The software is usually written in C, with the help of an entirely realized C compiler and only in a small number of cases is written at assembly level. The comfortable developing on foreign platform (usually PC) is typical. The programme can be loaded into the microcomputer via an intelligent interface, and/or electronically, or can be burned with masking technique.

The Cross-Development Tool is typical an Integrated Development Environment (IDE) which is fullintegrated in the Cross-operating system. The IDE contains case-sensitive source-text editor, macroassembler, powerful symbolic debugger with trace functions, advanced software cross-simulator for a wide range of devices (included peripheral simulation). One of the widely spread system is the Microchip's MPLAB (Fig 5.)

The software development methods in this range are really not supported with widely spread standard methodology.

C. *Typical software development methods of new type, high performance embedded systems*

The new applications, mainly in the field of mobile informatics and multimedia, having developed due to improvement in tools established new demands and expectations required from software products and thus from developing methodologies. These expectations are high from both the user's and the application environment's side, being present in three fields:

1. From the user interface side the users have already accustomed to the comfortable Windows-based interface in case of PCs and would like to see the same in their small-sized devices as well. The windows-based graphical interface is, however, resource intensive and requires the development of significantly more complex programming.
2. The processing speed is a determining feature partly for users (do not want to wait a lot) and it is for the application environment as well (realtime applications).
3. The adequate solution for multitasking is also a significant claim, since the majority of these systems execute realtime commands. The realisation of all these is a serious challenge in case of scarce resources. (eg. use of cellular phones)

All three requirements necessitate the development of much more complex systems, which require the strict usage of developing methodologies and thus an indispensable paradigm change widespread and in the category of high performance embedded systems in general.

On examining the software development options the origo is not to make the solutions applied for embedded systems extraneous for the world of PCs since the development must be conducted there. Another significant aspect is that it is much easier to phase in and spread a well-known already introduced methodology, phylosophy in the world of micros since the user willingness is larger.

At the selection of the methodology and the application environment the following aspects are practical to be considered:

- To be a well-known, acknowledged and widely used methodology
- To ensure large machine, PC support
- To ensure comfortable developing environment
- To enable real-time task management
- To realize the most possible out of the object-oriented concepts
- To ensure adequatly large programme directory
- To produce extremely compressed code since memory capacity is restricted in the micro environment
- Not to require large resources in process, because then the running of the application slows down
- To ensure portability of the application, namely to make the runner programme accessible on most physical architectures (machines)
- To ensure platform free running, namely not to bind to any attributes of physical or operating systems

In accordance with the statements in chapter II, a logical step is the use of UML in developing embedded systems. Apart from some benefits, this solution has some drawbacks as well:

- The developers of embedded systems found UML a too large and laborious system compared to the environment used so far.
- In the beginning, UML did not have such techniques with which real-time solutions could be effectively managed in case of embedded ystems.
- Such Tools were missing that generated applications to real-time operating system environment.

IV. UML-RT – The New Way to Software Development for High Performance Embedded Systems

In the last years UML has recently undergone a major upgrade, resulting in a revision called UML 2. The motivation for this revision was to make UML better suited to model-driven development MDD, and for embedded systems. So in order to eliminate the previously described disadvantages, the real-time UML (UML-RT) has emerged.

UML-RT is an extension of standard UML for better modeling embedded real-time systems. UML-RT adds new elements to the standard UML model. Three of them are used to model the architecture of the system (capsules, ports and connectors) and the fourth element (protocols) to model the communications within the system [4]. The new blocks serve the following purposes [5]:

Capsules: model the complex components of systems. Capsules may be structured hierarchically, containing subcapsules in it. Depending from the abstraction level, subcapsules may contain further subcapsules.

Ports: are like protocols which define the information-flow between the capsules and the environment. Ports can be private or public. Public ports have to be located on the border of the capsules.

Connectors: are used to interconnect two or more ports of capsules and define the communication relationships between capsules.

Protocols: are used to specifying the type of interactions between the connectors (for example a communication sequence).

With the application of UML-RT in embedded systems several paradigms have already been elaborated an tried out.

REFERENCES

[1] R. Hartenstein, "The changing role of computer architecture education within cs curricula." Invited talk, Workshop on Computer Architecture Education WCAE 2004 at 31st International Symposium on Computer Architecture. http://helios.informatik.uni-kl.de/staff/hartenstein/lot/hartensteinwcae04.

[2] Gy. Györök, "Embedded hybrid Controller with programmable Analog Circuit." IEEE 14th International Conference on Intelligent Systems. Gran Canaria, Spain, May 5-7. 2010., pp. 1-4., Paper 59., ISBN:978-4244-7651-0

[3] Gy. Györök, "Sensorless DC Motor Control." 11th International Carpatian Control Conference. Eger, Hungary, May 26-28. 2010., pp. 71-74., ISBN:978-963-06-9289-2

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

[5] C. K. Duby, "Accelerating Embedded Software Development with a Model Driven Architecture." Pathfinder Solutions, September 2003.

[6] P. Marwedel: "Embedded System Design." Kluwer Academic Publishers, 2003., ISBN-10 1-4020-7690-8

[7] A. Sangiovanni-Vincentelli, G. Martin: "Plattform-Based Design and Software Design Methodoogy for Embedded Systems", IEEE Design & Test of Computers, November-December 2001, pp. 23-33.

[8] R. Orfali, D. Harkey: "Client/Server Programming with JAVA and CORBA", John Wiley & Sons, Inc. New York, 1997, ISBN 0-471-16351-1

[9] N. Wirth: "Compiler Construction", Addison-Wesley, 1996, ISBN 0-201-40353-6

[10] Andrew W. Appel: "Modern Compiler Implementation in JAVA", Cambridge University Press, New York, Cambridge, 1997, ISBN 0-521-58388-8

[11] B. Venners: "Inside the JAVA Virtual Machine", McGraw-Hill, 1999, ISBN 0-07-135093-4Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551,