Challenges in the Development of Mechatronic Systems: The Mechatronic Component

Kleanthis Thramboulidis, Member, IEEE
Electrical & Computer Engineering
University of Patras, Greece
thrambo@ece.upatras.gr

Abstract—There is a need to replace existing mechanical and electromechanical systems with new ones, where functionality will be mainly implemented by software. The traditional approaches in the development of this kind of systems, which are known as Mechatronic systems (MTSs), prove wholly inappropriate for today’s problems that are characterized by complexity, dynamics and uncertainty. In this paper, challenges in the development of MTSs are discussed and the Mechatronic Component is proposed as the key construct to address these challenges. The means that the construct of mechatronic component addresses the challenges of synergistic integration, size and complexity and reuse are discussed and the infrastructure for a prototype implementation is described.

Index Terms—Mechatronics, Mechatronic Component, Model Integrated Mechatronics, Component based development, UML modeling.

I. INTRODUCTION

The name Mechatronics has its origin in the 1969 when a Japanese company used it to define the integration of machanics (“mecha”) with electronics (“tronics”). During the next years the term has taken a wider meaning and a number of definitions have been proposed in the literature (http://www.engr.colostate.edu/~dga/mechatronics/definitions.html). One of the most commonly used definition emphasizes on synergy. According to this, Mechatronics is the “synergistic integration of mechanical engineering with electronics and intelligent computer control in the design and manufacture of products and processes” [1]. Another definition focuses on functioning and defines Mechatronics as these systems “whose function relies on the integration of electrical and mechanical components coordinated by a control architecture.” During last years the term was significantly changed mainly due to the increasing importance of software in today’s systems. A brief description of the evolution of mechatronics can be found in [2] where a wider definition for the term is also proposed. According to this, the term mechatronics can be used to describe a philosophy in engineering and not just the technology itself.

There is currently an overlapping between the terms mechatronic system and embedded system. An embedded system “often includes task-specific hardware and mechanical parts” as is the case of the hard disk of the personal computer. This is why a big influence from software engineering and embedded system’s engineering is expected into the mechatronic systems development process.

The development of electronic and software parts of mechatronic systems are currently dominated by embedded systems development processes and toolsets. However, according to the results of an inventory carried out on European companies in the embedded domain [3], “the development process is unsatisfactory and many years behind their desktop counterparts.” Authors of the same paper also argue that “currently used development technologies do not take into account the specific needs of embedded systems development.” The common practice is that software development starts when the development of electronic and mechanical is already at a stage where any change in these parts is expensive and time consuming. This is why the mechanical and electronic properties impose a lot of constrains and narrow the solution space for software development. Problems that should have been addressed in the mechanical or even the electronic domain should be solved in the software domain. Even more, in some cases software is used to fix hardware and mechanical parts architecture design problems. This is among the main reasons for the proposed system not being the best solution for the specific problem.

Looking at the tools that are used for the development of such systems, we can notice that currently available toolsets are in the form of monolithic applications with proprietary interfaces that do not allow for tool integration not even exchange of development artifacts [4]. A great number of component models are under development but most of them specialize to a specific domain [4]. Their major constraint considering extensibility and interoperability is the fact that they have embedded the application model.

A convergence of mechatronic and embedded system domains is expected to take place in the near future. In this direction, the main objective of this paper is to identify a list of challenges in the development process of Mechatronic systems and propose a component based development process that will
address these challenges. Among the main challenges in the development process of Mechatronic systems we discriminate:

- synergistic integration;
- size & complexity; and
- reuse.

Basic concepts from the fields of reuse, architecture and component that can be used for the evaluation of development technologies are presented to provide the infrastructure for the definition of a development process that will address the above challenges. The paper was motivated by challenges that were identified in real world companies [5], which follow a component based development process. These challenges are:

- the number of components in a real company can grow exponentially;
- the effort to find the right component can be significant;
- any need to modify a component (even slightly) will vitiate benefits of having already tested components;
- it might be difficult to integrate the available components; and
- there can be several alternative designs (and interfaces) for one component.

The paper builds on the concept of Model Integrated Mechatronics (MIM) [6] and extends and further details it to address the above challenges. Extensibility mechanisms of UML [7] are used to define the basic constructs required to create the Architectural model of the system in the form of UML component diagrams.

The remaining of this paper is organized as follows: Background work is presented in the next section. The traditional development process is described and the need for an integrated approach is identified. The example system that is used as a running example in this paper is introduced and related work is presented. In section 3, the first of the challenges, i.e., the synergistic integration is presented and a proposal to address it is described. In section 4, the size and complexity of Mechatronic Systems (MTSs) are considered and the role of the architecture is discussed. Reuse is discussed in section 5, a prototype implementation in section 6 and finally the paper is concluded in the last section.

II. BACKGROUND WORK

A. The traditional development process

According to the traditional approach mechanical, electrical, electronic and software parts that constitute the system are developed independently, in an end-to-end fashion and then are integrated to compose the final system.

For example, for the Festo Modular Production System (MPS) that is used as a running example in this paper, the development of the mechanical process, namely the Festo MPS Mechanical Process, was carried out independently of the development process of the controlling system. Festo MPS is a well documented system used by many universities for research and education [8][9]. As shown in the schematic diagram of figure 1, the system is composed of three units.

Each one of the three units that constitute the system, i.e., the distribution unit, the testing unit and the processing unit, is further defined as aggregation of other mechanical units.

The distribution unit, which is composed of a pneumatic feeder and a converter, forwards cylindrical work pieces from the Stack to the Testing unit. The testing unit is composed of the detector, the elevator and the shift out cylinder. The detection unit performs checks on work pieces for height, material type and color. Work pieces that successfully pass this check are forwarded to the rotating disk of the processing unit, where the drilling of the work piece is performed as the primary processing of this MPS. The result of the drilling operation is next checked by the checking machine and the work piece is forwarded for further processing to another mechanical unit. A detailed description of Festo MPS can be found in [8] and [9]. More information on the Festo MPS case study can be found at http://seg.ece.upatras.gr/seg/dev/FestoMPS.htm. Alternatives for the design of the controlling system using a component based approach are given in [10].

B. The need for an integrated approach

Fig. 2 shows the adopted architecture for the Festo MPS according to the traditional approach. According to this approach even the constituent components of the controlling subsystem, i.e., the control application and the communication and processing subsystem, are developed independently and then are integrated to compose the final control system.
This is the approach that was adopted in the early stages of the CORFU project [11]. Corfu is a framework for the development of the software part of mechatronic systems following the IEC61499 Function Block model. However, the approach is quite general and can be adopted to support other component models. Corfu is based on a 4-layer architecture (4LCA) that was our first attempt to clearly identify the different constituent components of manufacturing systems and capture their relationships. Later on, from the enhanced-4LCA, it became clear that a high-quality design for the whole system could not be achieved without simultaneously considering the engineering process for all constituent parts, which is also a necessary precondition for system simulation [6]. This was the motivation for the integrated-component approach we next adopted.

From the experience gained all these years working with the development of such environment, i.e., the CORFU framework and the 4LCA, it was clear that materials handling, energy conversion and information processing that constitute MTSs, interact in many complex ways. This makes a high-quality design for the whole manufacturing system not feasible without simultaneously considering the engineering process of all constituent components. Rzevski [12] claims that “current mechatronic design practices are wholly inappropriate for problems characterized by complexity, dynamics and uncertainty, as today Mechatronic systems.” Amerognen [13] states that “changes in the construction and the controller is important to be evaluated simultaneously during the design” and also that “a badly designed mechanical system will never be able to give a good performance by adding a sophisticated controller.”

This problem is analogous with the Data Structures and Algorithms problem that the software community confronted many years ago. It was found that even with the best algorithm the result was not the optimal one if the corresponding data have not been properly organized. The concurrent engineering of data structures and algorithms is the prerequisite to get the optimal solution. This is also analogous with the problem that was recognized last years in the embedded systems’ domain regarding the design of software and hardware. It is now widely accepted that co-design, i.e., the simultaneous design of both hardware and software, is the approach selected to optimally solve this problem. We strongly believe that this model will be extended very soon to cover the mechanical part of complex mechatronic systems.

C. Related work

Other researchers are already working in the direction of improving the effectiveness of the development process of Mechatronic systems. Habib [2] argues on the urgent need for theories, models, and tools that should facilitate modeling, analysis, synthesis, simulation, and prototyping of multitechnological systems such as Mechatronic systems. He emphasizes the argument that the approach based on optimization within each domain separately will not result in the optimum system design, and he proposes a data and model integration approach to address the integration problem. Knor et al. [14] briefly refer to a process model of Robert Bosch GmbH for the development of mechatronic systems in Motor Vehicles to support aspects such as re-use, exchangeability, scalability and distributed development. They use the concept of Mechatronic Component, even though not well defined, to define the basic construct of their process. They argue: a) the need of a clear specification of component interfaces; and b) the great contribution of re-use to increase the quality properties of mechatronic systems and decrease development time. Nordmann [15] is using the concept of mechatronic component and presents an example of using Active Magnetic Bearings (AMB) to increase performance, reliability, reusability and longer lifetime. El-khoury et al. [16] propose the development of multidisciplinary systems such as mechatronics the integration of the various domain-specific tools. The main focus is on the integration of used data and models and not on the process level.

The extensibility mechanisms of UML are used to define profiles or meta-models on specific application domains. An example, of such a profile is the one presented in [17] where a UML profile for applications in the automation industry is introduced. It should be noted that it is not an objective of this paper to introduce such a profile but just to propose the use of the MTC in the architectural design phase, to handle the increasing size and complexity of MTSs.

Authors in [18] propose an extension of UML for modeling Mechatronics systems. ‘Mechatronic UML’ is defined as an extension of UML to built platform independent models for Mechatronics systems. Various UML models have been extended to cover the requirements of modeling the structural view as well as the behavioral view of the system. However, the proposed extension is used to model only the software part of the Mechatronic system.

III. SYNERGISTIC INTEGRATION – THE MECHATRONIC COMPONENT

A. Towards a Synergistic Integration

We categorize the development approaches to improve the effectiveness of the development process in the following categories:
1. Implementation time integration.
2. Design time integration at data and model level.
3. Design time integration at process level.

1) Implementation time integration.
This approach is the one adopted by the traditional development methodologies. According to this the constituent components, i.e., the mechanical, the electronic and the software are developed independently. All the implementation level constructs are integrated in a subsequent phase. To improve the effectiveness of the development process, this approach focuses on increasing the competence in each discipline.
2) Design time integration at data and model level. This approach is slightly incorporated in the current development practices, attempts to get a better result by improving design level synchronization [2]. This can be achieved either in the data level, through a common data repository used from the different domains or by a common model with which the different domains' models should be compliant. The ability to interchange models between design tools from different domains is of great importance to this approach [19].

3) Design time integration at process level. This approach focuses on the integration between the design processes of the different domains. According to this, the sub-optimal design resulting from each engineering domain is not enough to lead to the optimum system level solution. To overcome this limitation, the approach proposes the concurrent development of different components of various disciplines and proposes a design level integration at the component [6] or module [20] level, with emphasis on optimizing in the component or module level, rather than in the discipline level. This approach accepts the concurrent development of different components of various disciplines and proposes a design level integration at the component level. It defines the concept of the component or module and focuses on the definition of the development process. In fact, Mechatronic components already exist in the market, e.g., the ones shown in figure 3. These Mechatronic components are used in the development process of MTSs.

Fig. 3. Real-world Mechatronic Components.

B. The Mechatronic Component

The MechaTronic Componnet (MTC) was defined as key construct in the context of the MIM architecture [6]. It is claimed in this section that the MTC addresses the requirements for synergistic integration in the process-integration level. A MTC is composed of a mechanical part, an electronic part, and a software part. However, the level of granularity for the concurrent engineering of constituent parts is still not yet defined. A number of parameters will probably be used to define this level and of course this will be domain specific.

For the definition of the term MTC the various definitions for the term component in software engineering were examined. The definition given in [21], which is the one finally adopted as basis for the definition of MTC, is briefly presented in the following.

Szyperksi [21] to get the right definition of the term “component” in the context of the component based development, defines four ‘tiers’ of reuse. These tiers are:

- make-and-buy 'tier';
- built-time components ‘tier’;
- deployable component ‘tier’; and
- the dynamic ‘tier’.

Szyperksi argues that practical use of components today covers partially the third tier which is mainly used for the definition of the term component. The deployment tier assumes that components from multiple vendors are integrated on site and not as part of the software construction process. He defines the customer-driven integration as deployment and the matching components deployable components. The main characteristics of deployable components are:

- a) they remain identifiable in the deliverable system;
- b) they are units of deployment; and
- c) they are ‘real’ software components.

It should be noted that deployment is defined as the last step where detailed configuration and customization for components are allowed. According to the above approach in the definition of software component, a MTC has to be

- a unit of deployment,
- a unit of versioning and replacement.

This means that a MTC cannot and should not be modified before its use; it may only be parameterized for the specific context. An MTC should be defined in such a way so as to hide its implementation behind a set of external interfaces. An MTC should constitute the basic building block for the development of MTSs. MIM, that promotes process level integration in mechatronic systems’ development, defines the MTC construct. At the same time it attempts to address other challenges in Mechatronic development, such as:

- the definition of an architecture for the concurrent integration of interdisciplinary domains engaged in mechatronic system; and
- the definition of a clearly specified interface description language for the MTC-interface description.

An MTS is defined, according to the MIM, as an aggregation of interconnected MTCs. MTCs satisfying the same interfaces may be substituted freely as is the case of the hard disk. This MTC has complicated software and electronic interface but it has a very simple mechanical interface that is its mounting support.

To model the development process of MTSs we defined the Mechatronic Layer (ML) and divided this in three spaces that correspond to the Analysis, Design and Implementation phases of the development process [6]. The development process in ML is very complex since it is an information integration process that crosses the boundaries of mechanical, electrical and computer science fields. To simplify the development process we project the artifacts of this layer (use cases, components, connectors, etc.) to the three layers below to get the corresponding constituent components that encapsulate the
implementing it. According to the above the architecture of the works, and allows the analysis of system’s properties without the system, provides a better understanding of how the system of a well documented architecture, reduces the complexity of organized structure of a software system expressed in the form known as ‘Programming in the Large’ paradigm. A well-provided the architectural and structural view of a system is modules that constitute the system. This latter paradigm that system structure and on the dependencies between different it is more efficient to focus on the design and specification of can also be used for the same purpose. A port is defined as a connection point between a MTC and its environment. A port is defined by a set of required and/or provided interfaces and optionally includes for the software part an attached behavior specifying the interactions through the software part of the port. Mechatronic Components are connected through ports according to provided and required interfaces declared on a port.

**IV. SIZE AND COMPLEXITY**

The size and complexity of the embedded software are increasing rapidly. ES roughly follows Moore’s law, doubling in size every two years. The ‘Programming in the Small’ paradigm that is widely used in the development process is proved inadequate to address the always increasing complexity of today’s ESs. This paradigm has been proved efficient to solve the question of how to describe algorithms and the corresponding data structures. However, for complex systems it is more efficient to focus on the design and specification of system structure and on the dependencies between different modules that constitute the system. This latter paradigm that provides the architectural and structural view of a system is known as ‘Programming in the Large’ paradigm. A well-organized structure of a software system expressed in the form of a well documented architecture, reduces the complexity of the system, provides a better understanding of how the system works, and allows the analysis of system’s properties without implementing it. According to the above the architecture of the MTC should be an abstract representation of the system in terms of its subsystems or components. This representation should suppress the details of elements that do not affect the way they are used by, use or interact with other elements of the system. It will be expressed in the form of:

- elements, that are called MTCs;
- the externally visible properties of those elements, that are known as interfaces; and
- the relationships among them that are known as ports and connectors.

Interfaces are in the form of:

- provided interfaces that describe the functionality that is supported by the component; and
- required interfaces that describe the functionality that the component needs from other MTCs.

**Fig. 4. The TestingUnit Mechatronic component**

For the analysis of system’s properties a textual representation of the MTC in required. Such a representation is given below in Acme like syntax. Acme is a simple, generic language for describing software architectures defined at Carnegie Mellon University [22]. An XML based description can also be used for the same purpose.

```java
MTS TestUnit = {
    MPort testWorkPiece;
    MPort legalWorkPieceOut;
    MPort illegalWorkPieceOut;
    MPort powerSupply;
    property pieceTestingTime:Float = 2.0
    <<units = "sec">>;
    property MechanicalPartSimulator:
        File = 'URL', Type = IEC61499 FB;
    property SoftwarePartImplementation:
        File = 'URL', Type = IEC61499 FB;
}
```
MTCs are connected using Mechatronic Connectors (MTConnectors) which are used to represent the communication glue that captures the nature of an interaction between MTCs, as shown in figure 5. Two MTConnectors are used to interconnect the TestingUnit MTC with the ProcessingUnit and recyclingUnit MTCs. MTConnectors might define characteristics of a material, energy or information communication channel, a synchronization model for communication, a communication protocol, etc. A pipe is a classical example of an MTConnector that represents a material communication channel. A belt can also be modeled as an MTConnector. Modeling the pipe and belt as MTConnectors extra behavior and intelligence can be appended to classical pipe to get extra functionality compared to the poor material transfer. In this case pipe and belt will have their own processing unit on which the basic software that monitors and/or controls the pipe or belt would be executed.

Figure 5. MTConnectors are used for the integration of MTCs.

Figure 6 shows the architecture of the Festo MPS adopting the proposed notation. The system is described as a set of components and connectors, describing the elements of the MTS and how they interact. Connections to the external components recyclingUnit and XUnit are also shown. This model can also capture properties that describe system-level attributes.

Synchronization between the different units of the system can be realized either using mechanical, electronic or software solutions. The trend is to realize the control and synchronization mostly in software. This is also true for the reconfiguration functionality. For example the synchronization between the DistributionUnit and the TestingUnit is implemented in our prototype implementation for the Festo MPS using a software solution. The MTConnector that interconnects the DistributionUnit with the TestingUnit has the proper software interface that allows the TestingUnit to inform the DistributionUnit that it is ready to accept a new work piece. It is clear that such synchronization can be implemented using an electronic but also a mechanical solution. The same software interface exists in the MTConnector that interconnects the TestingUnit with the ProcessingUnit.

It is clear from the above that the proposed modeling approach for the structure of the system can be used for the modeling of both traditional and new generation Mechatronic systems. If for example we consider the break system of an automobile we can discriminate five main components, i.e., the four break-calipers and the break pedal. These components are interconnected though MTConnectors that have, in the traditional system, the form of mechanical linkages and hydraulic pipes. This means that the constituent components of the traditional automobile break systems communicate through mechanical and hydraulic interfaces that define the components’ ports. A more modular architecture with lower coupling between its constituent components is the one proposed by the break-by-wire approach. Constituent components have no mechanical interaction; they communicate through ports that define electronic and software provided and required interfaces. MTConnectors are used to interconnect ports with compatible electronic and software interfaces. This leads to a modular mechatronic architecture that has the potential of removing mechanical linkages leading to more freedom in the design of the vehicle and also faster, and less complicated, assembly as is also claimed in [20].

From the above definitions on the system’s architecture we can formulate a number of questions that will help in the evaluation of development technologies of mechatronic systems. Some of these questions are:

Does the technology under evaluation provide constructs that help to understand the system at a high level of abstraction?
Does it provide some way of capturing system-level structure in terms of large granularity components?
Does it abstract away from “implementation level” details?
Does it support basic architectural constructs such as system, component, connector, port, and role?
Does it allow the system to be presented in such a way as to meet its behavioral, performance and life-cycle requirements?

V. REUSE

Reuse is one of the most important challenges in software development. One of the most important objectives of every methodology, technology, toolset is to increase reuse. The final objective of reuse in software development can be captured in the following sentence: “Make something once and reuse it
several times”. Even though this sounds easy, in practice it has been proved very difficult to apply. The following questions that represent a great challenge for reuse have to be answered:

- What to make?
- How to make?
- Who is going to make?
- How to reuse?

In this section the MTC package is proposed to be the unit of reuse in Mechatronic systems. An MTC package is defined as an aggregation of all these artifacts and knowledge which are required for the proper use of a mechatronic component in the development process. The following artifacts constitute a MTC package:

- the real world MTC,
- the simulation model of the MTC,
- metadata (service interface, QoS, service semantics, etc).

A. The MTC Construction process

MTC vendors should address in the development process of the corresponding MTC package the following well known from software engineering challenge related to reuse: “Produce that product, and on the way, produce components too”.

In fact MTC development is a very complicated task. The MTC developer should try to generalize the particular problem to be solved by looking for some more general problem analogous to this specific one. It is evident that not only the component itself must be generalized, but also the approach for the test of the component should address the generalized problem [23]. This means that the complexity in MTC development arises from requirements, design, coding, and testing. The following two “rules of three” for component reuse [23][24] should be considered as the starting point for the corresponding MTC components rules. The first one considers the effort needed to build reusable components and is defined as: “It is three times as difficult to build reusable components as single use components.” The second refers to the approval of the component as really generalized artifact: “a reusable component should be tried out in three different applications before it will be sufficiently general to accept into a reuse library.” Due to the complexity in MTC development and use, candidate components in MTS development should be MTC packages, which should have been documented and certified.

B. The MTC Acquisition process

As MTC acquisition is defined the process of obtaining an MTC. An efficient process for the acquisition of MTCs has been defined. MTC vendors construct the MTC package and publish it on the web. Semantic web provides an ideal platform for a flexible acquisition process. Such a process have been described and a prototype implementation has been presented to demonstrate the effectiveness of the process [25].

The solution is based on a fully distributed repository of MTC packages. Web services and Semantic Web through the use of ontologies provide a very promising approach for MTC acquisition which provides answers to the following challenges that were mentioned in introduction: “in a real company, the number of components can grow exponentially” and “the effort to find the right component can be significant”.

C. MTC Modification process

For the MTC modification the following rule of thumb from software component reuse should be considered: “If more than 20 to 25 percent of a component needs to be revised, it is more efficient and effective to rewrite it from scratch” [26]. This is due to the fact that modification of reused code is particularly error-prone. Regression testing is one solution used to address this problem. To avoid MTC modification, developers should consider as an alternative the reuse the solutions provided at the design level. This is exactly what design patterns in software engineering [27] try to address.

D. The MTC Integration process

The “ease of integration” is one of the most important parameters of an Architecture. To provide an easy integration process the MTC have been defined to have two aspects. The first one defines the external face of a part of a system and the second one implements the functionality of this part. This modularity increases the reusability of both constituent components, i.e., the system frameworks and the components.

Two alternatives exist for the integration process. According to the first one, the system is considered as an aggregation of interconnected MTCs, while according to the second one, the implementation of the MTS is specified using slots that may accept MTCs satisfying specific mechatronic interfaces. Mechatronic interfaces of the slots may be described using abstract MTCs.

VI. THE DEVELOPMENT ENVIRONMENT

Archimedes System platform is an under development prototype engineering support system to demonstrate the applicability of the concept of Mechatronic component and the construction of the mechatronic system as an aggregation of interconnected MTCs. The system is developed on top of the General Modeling Environment (GME) [28] which supports the easy creation of domain-specific modeling and program synthesis environments that provides the generic functionality of graphical development environments. Archimedes is an attempt to realize the basic concepts captured by the MIM architecture and in this context it addresses the following reuse levels:

- Component level, with MTC as the basic reusable unit.
- Mechatronic level knowledge. In this level that is supported through the framework, the reuse of other mechatronic artifacts of the development process such as analysis and design artifacts is supported.
- Application Domain Knowledge level. This level is supported through the metamodeling and the model transformation process.
Challenges in the development process of today’s complex Mechatronic systems are discussed in this paper. The Mechatronic Component (MTC) is proposed to be used as the key construct to promote the synergistic integration of constituent parts of mechatronic systems and increase reusability in the construction process. The ‘modeling in the large’ paradigm was adopted to address the always increasing complexity and size of mechatronic systems. According to this, the system is defined as an aggregation of existing mechatronic components interconnected using proper MTConnectors. The proposed approach that meets the discussed challenges, is in the context of the Model Integrated Mechatronics (MIM) paradigm that is partially implemented by the Archimedes System platform.

VII. CONCLUSIONS

These ideas were first presented in the Intelligent Mobile Work Machines workshop organized by the Information and Computer Systems in Automation laboratory at Helsinki University of Technology, Sept 2007. I would like to thank the organizers and especially Seppo Sierla and Kari Koskinen as well as participants for their fruitful comments.

ACKNOWLEDGMENT

These ideas were first presented in the Intelligent Mobile Work Machines workshop organized by the Information and Computer Systems in Automation laboratory at Helsinki University of Technology, Sept 2007. I would like to thank the organizers and especially Seppo Sierla and Kari Koskinen as well as participants for their fruitful comments.

REFERENCES

[27] E., Gamma, R., Helm, R., Johnson, J., Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software” Addison-Wesley, Professional Computing Series, 1995