APPLICATION OF SOFTWARE FRAMEWORK TECHNOLOGY

TO AN ANTENNA POINTING CONTROLLER

G. Montalto [*], A. Pasetti[**], N. Salerno [*]

[*]*Alenia Spazio S.p.A., via Saccomuro 24, 00131 Rome, Italy*
[**]*P&P Software GmbH, Peter-Thumb Str. 46, D-78464 Konstanz, Germany*
g.montalto@roma.alespazio.it, pasetti@pnp-software.com, n.salerno@roma.alespazio.it

Abstract

Software frameworks are a reuse technology that makes *architectural* (as opposed to *code)* reuse possible. They have been successfully applied in the desktop and commercial arena but have so far been eschewed in the space sector. Alenia Spazio, as part of a drive to increase software reuse, have started a pilot project to develop the software for an Antenna Pointing Controller (APC) using framework technology. This work is done with P&P Software of Germany and takes as a starting point the AOCS Framework. The paper describes the experience acquired and the lesson learned applying the framework technology to the APC project. The first part gives an overview of object-oriented frameworks in general and of the AOCS Framework in particular. The second part describes the APC Framework, its close likeness to the AOCS Framework and the methodological and implementation issues that arose in the application of framework technology to an Antenna Pointing Controller.

1. Object-Oriented Software Frameworks

Software frameworks [Pre95, Fay99] promote the reuse of entire architectures within a narrowly defined application domain. They propose an architecture that is optimized for all applications within this domain and make its reuse across applications in the domain possible. Frameworks can also be seen as generative devices: they serve as a basis from which applications can be rapidly instantiated.

Figure 1 shows a typical view of an application generated from a framework. The shaded block represents the architectural backbone of the application. This is invariant in the application's domain and is provided by the framework. The unshaded blocks provide the application-dependent behaviour. They customize the framework by being combined with it during the instantiation process.

The essential property of a framework is its *adaptability.* A framework exists solely to be adapted to match the needs of a specific application within its domain. The points where adaptation takes place are called *hot-spots.* In an *object-oriented* framework, two adaptation mechanisms are used: *inheritance* and *object composition*. Although frameworks can and have been built using other adaptation techniques (such as Ada generics), the experience from the commercial sector is that inheritance and object-composition are the only mechanisms that offer sufficient flexibility to adapt the behaviour

encapsulated in a component to the needs of differing applications – and hence to make that component reusable.
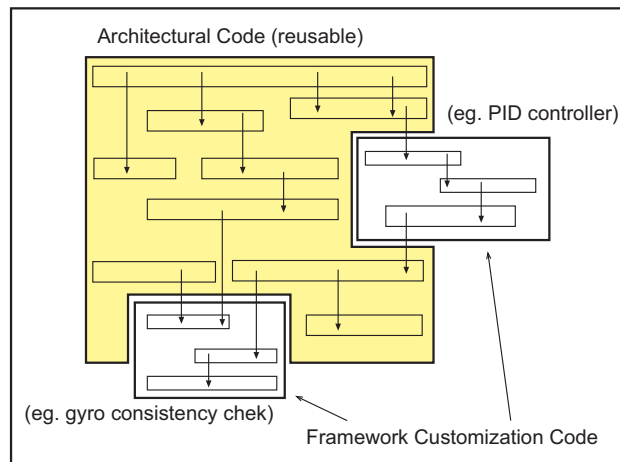

Figure 1: Structure of Applications Instantiated from the AOCS Framework

In practice, an object-oriented framework offers three types of constructs to application developers:

– *abstract interfaces* or sets of related operations without implementation
– *design patterns* or optimized solutions to recurring design problems
– *concrete components* or binary entities implementing one or more interfaces

The design patterns encapsulate adaptation mechanisms [Gam95] and are the vehicles through which architecture is made reusable. They are therefore at the heart of software frameworks. The combination of components and design patterns means that a framework allows reuse not just of individual components but of an entire system made up of the components together with their mutual relationships. Software frameworks are therefore much more than mere collections of reusable components because they offer components *and* their interconnections.

Software frameworks allow software reuse to take place at three levels as shown in figure 2. At *design level,* it is only the abstract design solutions embodied in the design patterns that are reused. At the next lower level, the reused elements are the definitions of the component interfaces and of their interconnections (namely *reuse of the architecture*). Finally, at the lowest level of reuse there is *code reuse* that consists in directly reusing concrete components or concrete classes. Traditional reuse technologies have been slanted towards code reuse. Software frameworks instead offer a more complete coverage of all three reuse levels.

The rational behind the innovative software reuse program started at Alenia-Spazio is the perception that traditional forms of reuse suffer from the following drawbacks:

– Reuse is *shallow* in that it is limited individual code fragments
– Reuse is *narrow* in that it tends to be focused on individual applications
– Reuse is *occasional* in that it has to rely on the same person or team being involved in two separate projects

The adoption of a framework approach allows all three above concerns to be addressed. Reuse becomes deeper because it is done at the level of architecture and design pattern as well as code. It also becomes broader because it becomes targeted at entire domains rather than just applications (a software frameworks models a set of related applications). Finally, reuse can be made more systematic because a framework can become a company asset that is applied to all projects in the framework domain.
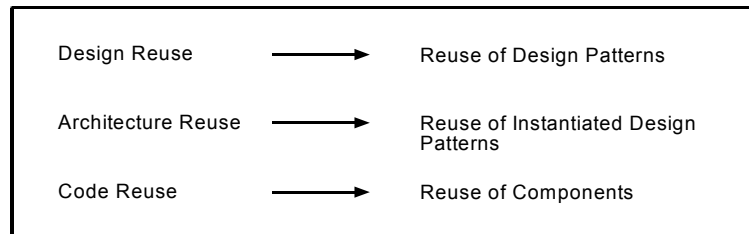
Design Reuse	⟶	Reuse of Design Patterns
Architecture Reuse	⟶	Reuse of Instantiated Design Patterns
Code Reuse	⟶	Reuse of Components

Figure 2: Levels of Software Reuse

2. The AOCS Framework

The reuse programme started at Alenia-Spazio aims at the development of software frameworks for the various domains in which Alenia-Spazio is active as a provider of software solutions. The starting point for these framework developments is the AOCS Framework. The AOCS Framework [Pas99, Pas01, Pas02a, Pas02b] is an object-oriented prototype framework that proposes adaptable solutions for the following functionalities:

– *Telemetry Functionality* (formatting and dispatching of telemetry data)
– *Telecommand Functionality* (management of telecommands)
– *Failure Detection Functionality* (management of checks to autonomously detect failures)
– *Failure Recovery Functionality* (management of corrective measures to counteract detected failures)
– *Controller Functionality* (management of closed-loop control algorithms)
– *Manoeuvre Functionality* (management of manoeuvre like slews, wheel unloading, etc)
– *Reconfiguration Functionality* (management of unit reconfigurations)
– *Unit Functionality* (management of external sensors and actuators)

Although the AOCS Framework was originally targeted at the attitude and orbit control subsystem only, the functionalities it covers allow it to be used for embedded control systems in general. The architectural solutions offered by the AOCS Framework are encapsulated in a set of domain-specific design patterns that are the heart of the framework itself. These design patterns are independent of each other and can be applied in isolation. Additionally, and in order to support reuse at architectural and code level, the AOCS Framework offers a set of abstract interfaces and concrete components that facilitate the instantiation of the design patterns for applications within the framework domain. The framework design patterns and abstract interfaces are obviously independent of the implementation language. The concrete components instead exist in a C++ and in a Java version.

3. The APC Domain

The APC (Antenna Pointing Controller) is the intelligent unit of an Antenna Pointing Subsystem (APS); it has in charge the management of all the subsystem activities. The APC unit, based on a DSP-21020, interfaces with the Satellite Control Unit (SCU), Tracking

Receivers (TRR's) and Antenna Pointing Drivers (APD's), as shown in figure 3, by means of two MIL-STD 1553B busses.
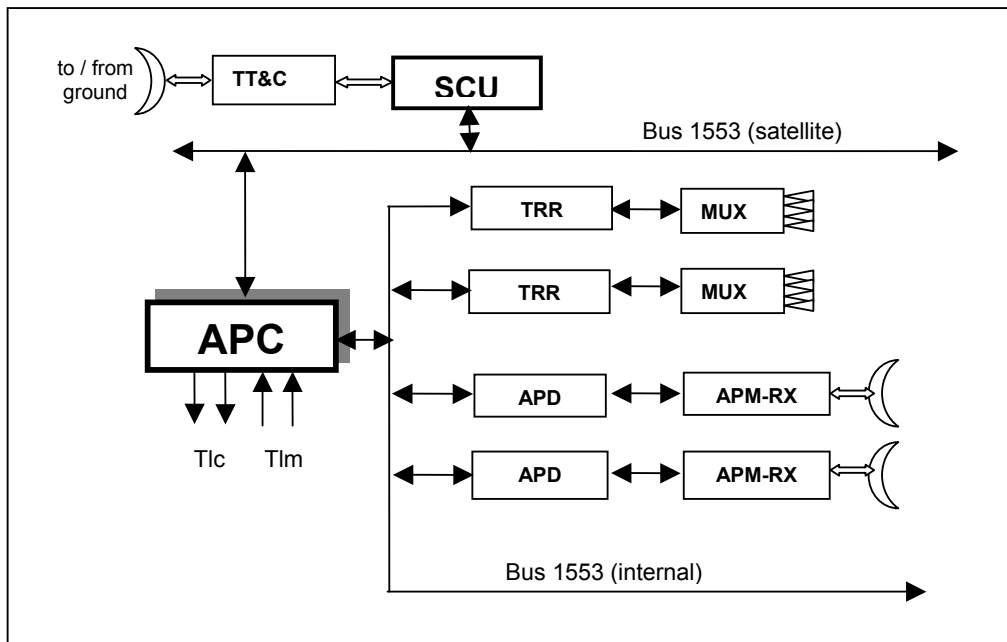

Figure 3: Architecture of an APC Subsystem

The DSP-21020 allows the software to perform its tasks. Moreover, it allows to handle TRR's sensor data acquisition and APD's commands transmission; in addition, it provides the direct interface between APC and TRR's/APD's units and other equipments, such as RF switches, to allow exchange of basic Tlc's/Tlm's (e.g.: direct commands to switch-on/off APD's and TRR's, telemetries carrying the powering status information, telecommands to RF-switches to reconfigure TRR's and so on). In details, the APC software is in charge to perform the following main functions:

1. Implementation of the operative modes to support the mission in all phases.
2. Implementation of the antenna control loop.
3. Commanding and managing of the antenna pointing mechanism (via APD's).
4. Data acquisition and managing of tracking sensors (via TRR's)
5. Exchange of TC/TM with SCU to allow the APC commandability and observability by ground stations.
6. Implementation of an FDIR strategy including automatic or ground-commanded mode transitions

The APC software implements the automa to handle the Operative Modes: Bootstrap (BOT), Init (INI), Antenna Control (CTR) and Emergency (EMG).

The BOT mode is entered immediately after an APC power-on/reset in order to perform the HW/SW initialization of the APS and to allow memory patch/dump from SCU; moreover, this mode puts the APC in a loop waiting SCU commands.

The INI mode is entered by means of an SCU command in order to perform other HW/SW initialization steps; then it switches-on or activates all controlled units (TRR's, APD's, MUX's etc.), according to the current configuration.

Also the CTR mode is entered by means of an SCU command. In this mode the APC is ready to start the automatic control of each antenna, according to proper control loops. The APC software performs, during this mode, the following functions:

− acquire data from the TRR's and APD's and pre-process them;
− compute the control logic algorithms foreseen by the current mode for each antenna;
− post-process and deliver commands to APD's;
− perform diagnostic of APS units and start recovery actions in case of anomalies;
− accept Tlc's / receive Tlm's from APS units.

The EMG mode is entered when an alarm is notified by SCU requiring a reduction of power consumption. In this case the APC software stops all automatic antenna control logic functions and performs a shutdown procedure, waiting for the APC switch off performed by SCU by means of dedicated command.

4. The APC Software and the AOCS Framework

Alenia Spazio has a long history of developing on-board software products, primarily in the payload area. Although there is much similarity across different applications, which indicates the existence of a potential for reuse, tapping this potential has so far proven impossible. This is primarily because past efforts at reuse have focused on reusing individual software components. This is difficult to do in practice for two reasons. Firstly, because differences in development environments and platform characteristics make reuse of code impossible. Secondly, because even small variations in requirements result in changes to the code of the component that is to be reused, which in turn requires revalidating the entire component and hence makes the cost of reusing it comparable to that of developing it anew. Object-oriented framework technology promises to address both problems. On the one hand, by allowing reuse of abstract design solutions and of abstract architectures, frameworks allow reuse to take place at a level higher than code thus decoupling it from platform and environment characteristics. On the other hand, object-orientation offers mechanisms that allow the behaviour encapsulated in a component to be adapted without changing its source code.

Alenia Spazio has therefore decided to adopt object-oriented frameworks as a reuse technology for its future programs. Arguably the most straightforward way to implement this decision would be to develop a framework and then to test its effectiveness by using it to develop several related applications. This however would be both long and expensive. The selected approach was to take the AOCS Framework as a starting point and use it to develop a framework for the APC domain. Partial use of the AOCS Framework was possible because of the similarities between antenna pointing control and attitude control subsystems and because the AOCS Framework design patterns are specifically designed to be used in isolation from each other thus allowing the framework to be used in piecemeal fashion. In an initial phase, the application of the framework approach is being restricted to only a subset of the APC software. This is primarily because the first application to be instantiated from the APC Framework, the APC software for a geo-stationary platform, will be written in C and implementation of object-oriented constructs in

C is awkward. A restricted application of framework techniques was also advisable for reasons of risk minimization.

Alenia Spazio is being supported in the use of the framework technology by P&P Software. P&P Software is a start-up company set up by the designer of the AOCS Framework that specializes in the application of object-oriented and framework technology to space projects. P&P Software makes the AOCS Framework available as a research prototype on a GNU general public licence.

The subset of the APC software that has been targeted by the framework evaluation exercise includes, mainly, the following functionalities: the management of telecommands, the execution of recovery actions, and the monitoring of variables (telemetries) for failure detection purposes. For all these functionalities, design patterns have been developed by suitably modifying the corresponding design patterns already offered by the AOCS Framework. Each design pattern is supported by a set of abstract interfaces and concrete components.

5. Methodological Issues

The adoption of a framework approach requires a radical change in the software design methodology. Traditional design methodologies implicitly assume an *object-based* application. By this it is meant that the software system is built as a collection of objects that encapsulate data while exposing operations to handle them. The design process typically starts with an analysis of the target application to identify *concrete* classes and *concrete* objects.

Use of framework techniques requires a different approach where the emphasis is on the identification of *points of adaptation* and of the *design patterns* and *abstract interfaces* that encapsulate the associated variability. Concrete components and classes are derived only after the design patterns and abstract interfaces have been defined. Note that traditional methodologies have no conceptualizations for concepts like points of adaptation, design pattern and abstract interface.

A further challenge lies in the need to reconcile the framework-based development process with the development process mandated by internal Alenia procedures and built around the "classical" software development process that starts with a URD (User Requirement Document), proceeds to an SRD (Software Requirement Document) and finally results in an ADD (Architectural Design Document) from which the detail code design is derived.

A discussion of the methodological approach adopted by Alenia-Spazio and P&P Software must be divided in two parts, one dealing with the methodology for the development of the framework, and one dealing with the development of applications instantiated from the framework. This section gives an overview of the development process used in the APC Framework project for the development of the framework and for the development of a geo-stationary platform application that is being instantiated from the framework.

The design process for the APC framework is based on that proposed in [Pas02] and is sketched in figure 4. The main change with respect to the development process of [Pas02] is the explicit introduction of a "Basic Operations" output.
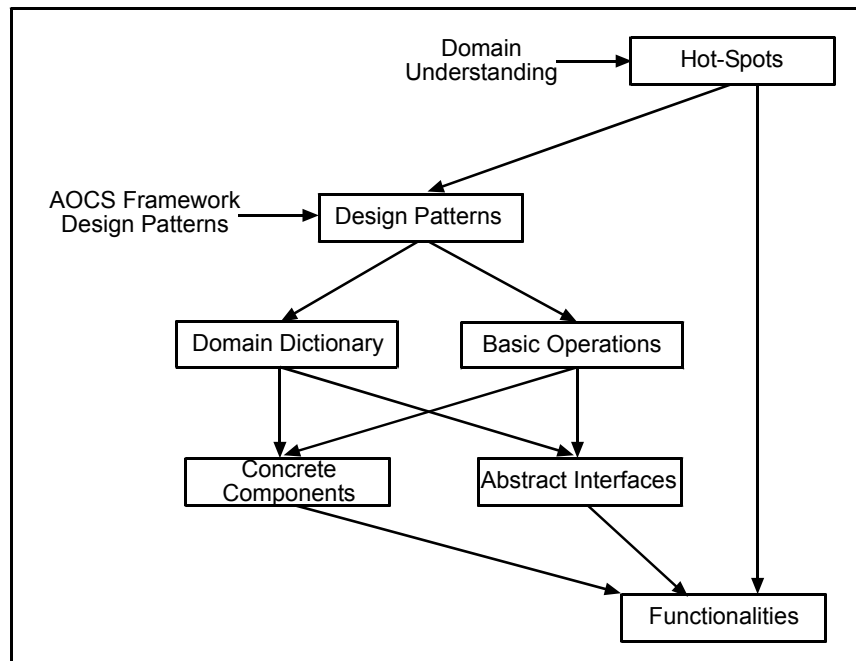
Figure 4. Framework Development Process adopted for APC Project

In the first step in the framework development process, the APC domain is analysed to identify requirements that are invariant in the domain and to separate them from requirements that are application-specific. The latter define the points where an APC framework must be adapted to the needs of a specific APC application. This leads to the definition of a set of *hot-spots* for the APC domain which represent the points of adaptation of the framework. In practice, this step was performed by analyzing a concrete but typical URD for an APC application and through discussion with domain experts in Alenia.

The variability inherent in a hot-spot is modelled by a design pattern. Hence, the second step in the development process is the definition of *design patterns* that are associated to the hot-spots identified in step 1. In the case of the APC project, the design patterns are obtained by adapting design patterns from the AOCS Framework.

The design patterns define a set of *domain abstractions* that encapsulate domain-wide concepts that can be used to express the framework design. The third step is accordingly the definition of a *domain dictionary* that defines the domain abstractions for the APC domain. The domain abstractions must ultimately be *mapped* to syntactical constructs consisting of *abstract interfaces* and *concrete core components.* The full definition of these constructs is performed during the framework architectural design phase. During the initial concept definition phase, only the *basic operations* of these abstract interfaces and concrete core components are identified. As the name implies, these basic operations define the essential services that are to be declared by the abstract interfaces and offered by the framework core components.

The framework development has a further final stage with the definition of the *functionalities* that describe the constructs offered by the framework from the point of view of prospective users and allow systematic matching with application requirements. Definition of the functionalities has not yet been done for the APC Framework.

7

The design process that is being followed to instantiate the APC software for a geo-stationary platform from the APC framework is summarized in figure 5.
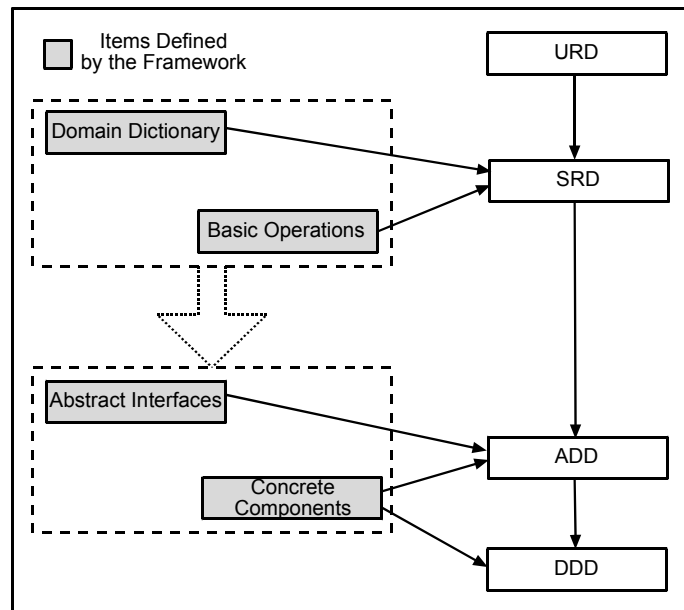

Figure 5: Application Development Process proposed for a geo-stationary platform

The impact of using the APC Framework is first felt when making the transition from URD to SRD. As the figure indicates, the SRD is written by taking into account the framework domain dictionary and basic operations. These define a set of terms that capture useful concepts in the APC domains. The software requirements should be written using, as far as possible, these pre-defined terms. These has three advantages:

− Software requirements that are more precise
− Software requirements that are more concise
− Software requirements that facilitate the definition of an architecture

The first advantage is a consequence of the use of terms that have a precise semantics defined by the framework.

The second advantage follows from the possibility of implicitly re-using in the formulation of the SRD the web of meanings that are defined by the framework. An example will clarify this point. Consider the *recovery action* abstraction. This abstraction is defined by the framework to encapsulate a generic response to the detection of a software anomaly during normal operation. The framework additionally defines a syntactical construct to which this abstraction is mapped (the `RecoveryAction` abstract interface) and it defines the basic operations attached to this construct. One of these basic operations is the enable/disable operation that allows the recovery action to be disabled and enabled.

Consider now an APC application where the user requirements identify a number of failure conditions and specify the responses that should be taken when a particular failure has been detected. Assume moreover that the user requirements also state that it should be possible to enable and disable the responses to individual failure conditions. The translation of these requirements from URD to SRD level can be easily done using the

framework abstractions by simply stating that the application should offer a number of *recovery actions* to be triggered in response to the detection of particular failure conditions. There is now no need to specify exactly what a recovery action is or what triggering it implies because all this is already defined by the framework. There is also no need to have a software requirement that it should be possible to selectively disable and enable recovery actions because this capability is already implicit in the recovery action abstraction as it is defined by the framework.

Finally, the third advantage to follow from the use of the framework terminology to express the software requirement is ease with which the application architecture can be derived. This is a consequence of the fact that the domain abstractions and the basic operations are mapped to concrete syntactical constructs. Thus, to take again the previous example, once the SRD states that a certain response to a failure conditions should consist of a *recovery action,* the application architect will use recovery action components (provided by the framework) to model the response to the failure.

At detailed design level, the concrete components provided by the framework can be directly incorporated in the application.

The above design process is being tested jointly by Alenia-Spazio and P&P Software in the development of the APC Framework and in its application to a geo-stationary platform. An important aspect of this experiment is that all framework documentation is being written in HTML and will be made available through the Alenia intranet. This will result in an HTML catalogue of design patterns and in an HTML version of the framework ADD. Both documents will be richly hyperlinked for ease of reference. The intention is to familiarize Alenia personnel both with the framework approach and with the specific abstractions proposed by the APC Framework. The long term objective is to bring users (the people who specify an application) to the point where they can directly use the framework and its abstractions and where they can write the URD in terms of these abstractions. This will help bridge the gap between software specification and software design and will shorten the software development process.

6. Implementation Issues

Object-oriented frameworks are best supported by object-oriented languages. Use of such a language in the geo-stationary platform case is not possible because of lack of compiler support for DSP-based targets, such as the APC unit where the geo-stationary platform application will run. It will be written in C and the object-oriented will be implemented in this language using the technique described in [Sam97]. This technique allows the implementation in C of constructs that are equivalent to C++ classes and to C++ objects. To each class, a C structure – the so-called Virtual Function Table or VFT – is associated that holds pointers to the functions representing the methods associated to that class. Objects (namely instances of a class) are implemented as pointers to C structures whose first field is a pointer to the VFT of the class from which they are instantiated. The implementation of the class and object structures is facilitated by the use of C macros that do some of the work that would normally be done by a C++ compiler. Note however that the lack of compiler support means that there is only very weak type checking and hence use of this technique requires considerable care.

In case of other systems implemented on the ERC32 or LEON processors the software development is fully supported for C++ which at least in the short term, is regarded as the ideal target language for object-oriented frameworks. Use of Ada95 is not considered because of the poor support that this language offers for object-oriented constructs (in particular, it is difficult to represent class hierarchies where classes implement several abstract interfaces). In order to reduce the risks connected to the lack of robustness of C++, Alenia-Spazio again with the support of P&P Software has started an internal project to define a safe language subset and robust coding rules that are targeted at the support of framework development.

7. Conclusions

Alenia-Spazio is currently involved in consolidating the software framework technology for the APC domain. This paper has described the technological and methodological aspects of these efforts and their expected benefits. A pilot project is targeting the APC domain. Experience to date has been very positive and Alenia-Spazio has started a new project to develop a software framework for transponder software which is another traditional area of work for the company.

8. References

[Fay99]   Fayad M, Schmidt D, R. Johnson R (eds) *Building Application Frameworks – Object Oriented Foundations of Framework Design*. Wiley Computer Publishing, 1995

[Gam95]   E. Gamma et. al., *Design Patterns*, Addison Wesley, 1995

[Pas99]   A. Pasetti, W. Pree, *A Component Framework for Satellite On-Board Software*, Proceedings of the 18-th Digital Avionics Systems Conference, St. Louis (USA), Oct. 99

[Pas01]   A. Pasetti, *et al, An Object-Oriented Component-Based Framework for On-Board Software*, Proceedings of the Data Systems In Aerospace Conference, Nice, May 2001,

[Pas02a] http://www.pnp-software.com/AocsFrameworkProject/ProjectHomePage.html

[Pas02b] A. Pasetti, *Software Frameworks and Embedded Control Systems*, LNCS Vol. 2231, Springer-Verlag, 2002

[Pre95]   W. Pree, *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1995

[Sam97]   Samek M, *Inheritance and Polymorphism in C*, Embedded Systems Programming, Dec. 1997