Composing ActivityCharts/StateCharts, SDL and SAO Specifications for Codesign in Avionics

M. Romdhani1,2  P. Chambert1  A. Jeffroy1  P. de Chazelles1  A.A. Jerraya2

1 AEROSPATIALE Aircraft, Avionics and Systems Division
316, Route de Bayonne, 31060 Toulouse Cedex 03, France

2 TIMA/CNRS-INPG-UJF Laboratory, System-Level Synthesis Group
46, avenue Félix Viallet, 38031 Grenoble Cedex, France

Abstract
This paper deals with the composition of system-level specifications, and more generally a multi-formalisms codesign methodology in the context of AEROSPATIALE Aircraft avionics systems. The methodology is based on a unified system model, named SOLAR, which is used to compose three specification languages: ActivityCharts/StateCharts, SDL, and SAO. The model serves the rest of codesign tasks. Multi-formalisms composition principles are illustrated on an avionics system that is part of AIRBUS A340 on-board systems family. Results and perspectives are also outlined.

1. Introduction
Today's avionics standards cover only the software development process. The aeronautical community is moving towards standardizing the hardware and the systems development processes[16]. These orientations promote towards structured methodologies that emphasize on systems modeling and allow for a joint hardware/software codesign.

Actually, the field of codesign itself is not recent, but the joint specification, design and synthesis of mixed hardware/software systems is a recent issue. Several projects currently in progress (SpecSyn at Irvine[5], CODES at Siemens[2], SDW at Italtel[1], Thomas approach at CMU[17], Gupta and DeMicheli approach at Standford[6], Ptolemy at Berkeley[10], RASSP[13], etc.) are trying to integrate both the hardware and the software in the same process. The proposed methodologies differ mainly from the following three key issues:

- the style of the input specification,
- the target architecture model, and
- the synthesis steps performed by the codesign system.

This paper deals with the first issue. Most existing codesign methodologies are based on a specific specification language. However, the specification of a very large design such as the electronic system of an aircraft may need the use of several specification languages.

The long term goal of this work is the definition of a requirements-driven codesign methodology in the context of AEROSPATIALE Aircraft avionics. The underlying motivations are enhancing the systems specification quality, optimizing systems performances and costs, and shortening systems development cycle.

Such a requirements-driven methodology should solve these two closely related problems:

1- Which formalism is the most suitable for specifying a given aspect of an avionics system?

2- How to compose partial specifications in order to achieve codesign?

The objective here is to integrate several specification environments used in avionics within a hardware/software codesign environment. The paper is organized as follows: First, the next section gives an overview of the codesign methodology, then section 3 deals with multi-formalisms specification in avionics. Section 4 details the multi-formalisms composition principles. Conclusions and future work are outlined in section 5.

2. Multi-formalisms Codesign Methodology
The AEROSPATIALE Aircraft's systems workshop is made up of methods and means covering the various development phases of avionics[3,14]. But, as it has been noticed in the telecommunications and DSP domains[10,19], the avionics software development process is currently achieved with no interaction with that of hardware. The specification formalisms are mainly used for validation and on-board code generation purposes.

2.1. The Codesign Methodology Flow
The proposed methodology is a system approach that starts from aircraft system requirements and leads to the first system prototype. It aims at a joint design of hardware and software and explores different architecture partitioning alternatives preserving the current qualified development

0-8186-7156-4/95 $4.00 © 1995 IEEE
methods and tools. An overview of the codesign methodology flow is shown in figure 1.

![Codedesign Flow Diagram](image)

**Figure 1: The codesign methodology flow**

The codesign flow consists of these steps:
- Multi-formalisms specification and validation
- Specification composition
- Iterative partitioning and performance evaluation
- Architecture prototyping

The specification step takes into account the use of environments that provide simulation and analysis facilities. In fact, when specifying, it is important to ensure that the system specification meets the requirements. Experience shows that detection of errors at this level saves much efforts and costs.

### 2.2. The System Model Structure

The system model, named SOLAR[9], is designed to accommodate various system specification languages such as SDL, LOTOS, or Harel Activity Charts/StateCharts. The objective is to allow the designer to use one or more of these languages and to translate these descriptions into SOLAR. The SOLAR model constitutes also the basis of the codesign tasks. It accommodates system-level partitioning and synthesis algorithms. The goal is to ease the automation of the design steps.

SOLAR is based on an extended Finite State Machine (FSM) model and allows several levels of description, starting from the level of communicating systems which contains a hierarchical structure of processes communicating via channels right down to the register transfer level and basic FSM descriptions. SOLAR's basic model is an extended FSM that allows the representation of hierarchy and parallelism. The main concepts are shown in figure 2.

### 3. Multi-formalisms Specification of Avionics

A new avionics development starts from several requirements expressed usually in a textual form. These include ad-hoc information about the system to develop: functions to fulfill, critical functions, performance constraints, costs, delivery delays, etc. A comprehensive analysis of these requirements should lead to structuring and formalizing the requirements. Textual information should be formalized as larger as possible in terms of functional and behavioral requirements.

#### 3.1. An Avionics Example: The AIRBUS A340 CMWC System

The CMWC (Centralized Maintenance and Warning Computer) deals with a virtual aircraft maintenance avionics system. It includes some functions of the AIRBUS A340 Centralized Maintenance Computer, known as A340 CMC.

![CMWC Diagram](image)

**Figure 3: The CMWC environment**

The CMWC is a highly reactive system. On one hand, it is connected to the man machine interface units. These are composed of a Multi-Control Display Unit (MCDU) and a printer. These units enable the pilot or the maintenance operators to initiate interactively maintenance tasks and to have personalized text reports about the aircraft systems. On the other hand, the CMWC interacts with all on-board avionics Built In Test Equipments (BITEs). These equipments transmit local warnings and faults information...
to CMWC and receive the maintenance orders. Figure 3 gives an overview on the CMWC environment.

3.2. The CMWC Multi-formalisms Specification

The global CMWC system specification is achieved through the ActivityCharts/StateCharts formalisms using STATEMATE® tool from i-Logix. Communication protocols are specified using SDL (Specification and Description Language), the International Telecommunications Union standard for protocols specification. The SDL tool is Géode from Verilog. The warning computation and maintenance management functions are specified using SAO (Computer Aided Specification). SAO is an AEROSPATIALE in-house visual formalism. It is based on a synchronous data-flow model and is usually used to specify the synchronous signals acquisition and processing. Table 1 summarizes CMWC specification formalisms. Figure 4 shows the CMWC multi-formalisms specification context.

<table>
<thead>
<tr>
<th>Function</th>
<th>Formalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMWC functional decomposition</td>
<td>ActivityCharts/StateCharts</td>
</tr>
<tr>
<td>CMWC operational modes</td>
<td></td>
</tr>
<tr>
<td>CMWC/MCDU, CMWC/Printer, CMWC/BITEs protocols</td>
<td>SDL</td>
</tr>
<tr>
<td>Warnings processing</td>
<td>SAO</td>
</tr>
<tr>
<td>Maintenance management</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The CMWC specification formalisms

The top-level functional model of the CMWC is produced using the ActivityCharts/Statecharts formalisms. The model is composed of several ActivityCharts modeling the system’s main functions and a control activity that coordinates the execution of the ActivityCharts according to the system’s dynamic operational mode[15]. The control activity is represented by means of a StateChart.

Model decomposition and refinement is done through a hierarchical approach according to the well known structured analysis method SA-RT (Structured Analysis for Real-Time systems)[7]. This approach enables the refinement of an activity either in ActivityCharts/StateCharts formalisms or in another formalism. The refinement of a system function corresponds, in some cases, to the re-use of an existing specification expressed in a different formalism.

4. Multi-formalisms Composition

4.1. Related Work

The multi-formalisms composition problem has been the topic of many researches mainly in the software engineering domain. Several approaches have been proposed in order to compose partial programming and/or specification languages.

Zave and Jackson’s approach[20] is based on the predicate logic semantic domain. Partial specifications are assigned semantics in this domain, and their composition is the conjunction of all partial specification. Wile’s approach[18] to composition uses a common syntactic framework defined in terms of grammars and transformations. Garden project[12] provides multi-formalisms specification by means of a common operational semantics.

These approaches are globally intended to facilitate the proofs of concurrent systems properties. Our interest is essentially focussed on specifications tool support and on the later codesign steps. The composition approach is based on underlying formalisms semantics and the use of the unified system model, SOLAR, which offers several abstraction levels. The completeness and the consistency proofs are the major problems of any multi-formalisms specification. In our context, these must be kept in mind by the specifiers and have to be solved immediately after the specification capture phase.

4.2. ActivityCharts/StateCharts Specifications Capture

The STATEMATE® environment[8] offers three distinct formalisms. (1) StateCharts are used to specify a system’s temporal behavior and control relations in a state-oriented model; (2) ActivityCharts for functional decomposition and information flow; and (3) ModuleCharts for physical structure decomposition. We have been limited on a joint use of StateCharts and ActivityCharts, since the ModuleCharts denote yet a physical architecture decomposition. However, this decomposition should normally be the result of codesign process.
Communication through shared variables could model the broadcasting scheme.

4.3. SDL Specifications Capture

SDL[4] is based on an extended FSM model. The main concepts handled by SDL are the system, the block, the process and the channel. The translation of most concepts into SOLAR, with the exception of communication concepts, is fairly straightforward.

The translation of the SDL communication concepts (Channel and SignalRoute) leads to a reorganization of the description. In SDL, communication is based on message passing. Processes communicate through SignalRoutes. Channels are used to group all the communications between blocks. The communication methodology used allows such systems to be modeled. This scheme is summarized in figure 6. Figure 6.a shows a system example, composed of two communicating blocks. Block b1 contains one process (P1) that communicates with P21 and P22, processes that belong to block b2. The translation of this system into SOLAR will produce the structure shown in figure 6.b.

Each SDL process is translated into a design unit composed of an extended FSM and a ChannelUnit. The ChannelUnit explicitly describes the SDL communication scheme. In SDL, FIFOs are implicitly included in each process, and all signals, no matter what their type is, are automatically stored in these FIFOs.

4.4. SAO Specifications Capture

SAO is a data-flow oriented formalism. An SAO specification is formed by a set of graphical sheets which describe transformational functions through the data flow model. The SAO environment uses a symbol library and simple assembly rules. Each symbol of the library has an associated algorithm to compute its output values. Figure 7.a shows an example of SAO sheets.
5. Conclusions and Future Work

This paper introduced a multi-formalisms specification codesign methodology in avionics. The multi-formalisms context discussed in this paper uses Harel's ActivityCharts/StateCharts as structuring formalisms, and SDL, SAO respectively, as specific formalisms for communication protocols and digital signal processing.

Composition of ActivityCharts/StateCharts, SDL and SAO specifications is based on the unified system model, SOLAR. We have noticed that SOLAR's extended finite state machine model facilitates the capture of StateCharts and SDL specifications since these are also state-based formalisms. The SAO data-flow specifications are simply captured as sequential procedures. SOLAR offers powerful communication abstractions that enable the capture of ActivityCharts/StateCharts broadcasting scheme, the SDL asynchronous communication scheme and the SAO discrete data exchange scheme.

Currently, a pathfinder for the methodology is in progress. Three translation tools have been developed to enable the capture of ActivityCharts/StateCharts, SDL, and SAO specifications in SOLAR. The translators act only on subsets of the specification languages and need to be extended. A complete SOLAR model has been generated for the AIRBUS A340 CMWC system.

A future possible support environment may be such as given in Figure 8. It is based on the SOLAR framework[9] for codesign and synthesis tasks.

Acknowledgement

This work is part of the CODESIGN project which is currently in progress within the AEROSPATIALE Group. The project is supported by the Aircraft Division, the Space & Defence Division, the Missiles Division, SEXTANT Avionique and EUROCOPTER-France. Authors thank all persons from these companies who discussed the ideas presented here and helped in preparation of the material of this work.

References


