Effective Cooperation of System Level and ECU Centric Tools within the AUTOSAR Tool Chain

Dirk Stichling, Oliver Niggemann and Rainer Otterbach
dSPACE GmbH, Germany

Karsten Hoffmeister
Elektrobit Automotive GmbH, Germany

Reprinted From: System Level Architecture Design Tools and Methods
(SP-2129)
ABSTRACT

The AUTOSAR methodology does not only specify XML file formats but defines a software development workflow for electronic control units (ECUs). This waterfall-like workflow does not cover the real process between car manufacturers (OEMs) and suppliers or even within these companies. Therefore productive AUTOSAR tools have to actively support these processes e. g. by providing an iterative workflow instead of a waterfall-like. One of the major topics within the processes is the coupling of system level design and ECU centric design. This paper evaluates the different use-cases between system level design and ECU centric design and presents the tool requirements which are necessary for productive work. The use cases are grouped into two major categories. First top-down, start with system design so that the system model defines requirements and interfaces of the ECUs’ basic software. Second bottom-up, the design of the ECUs with the I/O and basic services is fixed early, i. e. the basic software drives the design and interfaces of the system model.

INTRODUCTION

AUTOSAR [1], as a standardization organization, does not only specify file formats and basic software modules (operating system, communication stack, I/O, services, etc.), it also defines a corresponding methodology [2]. This methodology establishes new concepts like runtime environment, advanced error handling and new configuration concepts. A good technical overview of AUTOSAR is given in [3]. Traditional automotive software development tools are not well suited for the AUTOSAR methodology. New tools have to be introduced which are specially designed to be integrated in the AUTOSAR methodology. The AUTOSAR specifications only marginally consider the workflow between AUTOSAR compliant software development tools. That was the reason why dSPACE and Elektrobit set up a cooperation which addresses the challenges of an integrated AUTOSAR tool chain. Within this cooperation the requirements for an effective coupling of system level design and ECU centric tools were worked out and incorporated into the development of dSPACE’s new system level design tool SystemDesk and Elektrobit’s ECU centric tool tresos ECU. Software development processes of both car manufacturers and suppliers were evaluated. Two main workflows expose as a result of this evaluation: top-down and bottom-up. The presentation of these two workflows is the main aspects of this paper.

But first of all, the tasks of system level design and ECU centric design in respect of the AUTOSAR methodology are presented. Afterwards, dSPACE’s new system level design tool SystemDesk and Elektrobit’s ECU centric tool tresos ECU are introduced. The following chapter shortly defines the most important concepts of AUTOSAR. An explanation of challenges between system level design and ECU centric design is presented next, followed by two simplified contrary workflows, top-down and bottom-up. Each workflow chapter explains the fundamental principle of this workflow and how the challenges can be addressed by an AUTOSAR compliant tool chain. The chapter before last addresses some process related topics.

RELATED WORK

AUTOSAR uses the concept of a configurable and generated basic software core. This concept is not new to the automotive software development. Most OEMs have defined their own standard core comprising a OS (often OSEK-OS [4]), a communication stack, an I/O stack and basic software services. ECU centric tools like tresos
ECU are used to configure and generate this basic software for some years now [5].

Another key concept of AUTOSAR is the component-based approach for the specification of software architectures. This concept is not new as well. E.g. UML 2.0 [6] comprises component modeling means and in [7, 8, 9, 10] the trend towards a component-based design for automotive software is presented.

Nevertheless, the joint usage of basic software configuration and software architecture design is new. Therefore, new workflows and also tools are needed to support this new approach. The software architecture design based on the AUTOSAR methodology and dSPACE’s SystemDesk is presented in [11], the basic software configuration and generation based on Elektrobit’s tresos ECU is presented in [12].

AUTOSAR TOOLS

Figure 1 shows AUTOSAR tools, AUTOSAR file formats and their relationships. The following sections shortly introduce the different types of AUTOSAR tools:

BEHAVIOR MODELING

The upper part of figure 1 represents the development of reusable software components (SWC) either by using behavior modeling tools (BMT) such as Simulink or TargetLink, or manually written C code. In both cases each software component consists of a software component description in the form of an AUTOSAR software component description XML file and corresponding C code that uses the RTE API.

As an example, dSPACE provides an AUTOSAR module for the production quality code generator TargetLink. This module allows import and export of software component descriptions and it generates AUTOSAR compliant production quality C code. Nevertheless, design and development of application software components is not the focus of this paper.

SYSTEM LEVEL DESIGN

The software component descriptions are read and written by system level design tools. Within these tools software architectures of complete systems or subsystems are designed. This comprises tasks like assembling and connecting software components, specifying hardware topologies, handling network communication, mapping software components onto ECUs, mapping data elements onto network signals, etc. The result of system level design is a description of the overall system in the form of an AUTOSAR system description XML file. Additionally a system level design tool may export AUTOSAR ECU configuration XML files for each individual ECU. As a rule, this ECU configuration is not complete but contains only the parameters which can be derived from the system level.

SYSTEMDESK

SystemDesk is dSPACE’s new system level design tool (see figure 2). The main features of SystemDesk are:

- **Design of software architectures**: Diagram editors allow graphical design of hierarchical components comprising ports, interfaces and interface elements.
- **Formalization of hardware topology**: Editors allow design of hardware topologies comprising buses, ECUs and communication ports.
- **Design of network communication**: Communication matrices (e.g. DBC files) can be imported and timing requirements can be specified.
- **Design of systems**: One software architecture, hardware topology and network communication are assembled to form one system. Within a system, mapping of software onto hardware, mapping of data elements onto network signals, and ECU configurations are defined.
- **RTE generation**: AUTOSAR compliant RTE generation based on the production quality code genera-
tor TargetLink. The first release of SystemDesk will not support all RTE features but the most important ones, such as intra and inter ECU communication, task body generation, and usage of the AUTOSAR communication stack.

- **AUTOSAR:** Import and export of AUTOSAR software component descriptions; import and export of AUTOSAR ECU configurations; export of AUTOSAR system descriptions.

---

**ECU CENTRIC DESIGN**

Each individual ECU has to be configured and finally the HEX file has to be generated. ECU centric tools are used for this purpose. They read system descriptions and extract those aspects which are relevant for the corresponding ECU. Additionally they read and write the ECU configuration files which contain configuration information about all basic software components of one ECU including the RTE configuration. ECU centric tools generate C code of basic software components after the ECU has been configured successfully and all validation checks are done. In some cases ECU centric tools have to export software component descriptions of basic software components as well.

**TRESOS ECU**

*Tresos ECU* is the further development of Elektrobit’s *tresos* development suite (see figure 3). It comprises implementations of all basic software modules defined by AUTOSAR along with an advanced configuration and generation tool. Furthermore, *tresos ECU* includes an AUTOSAR compliant runtime environment generator. The main features of *tresos ECU* are:

- **One environment:** Within one tool environment all basic software relevant configuration information is handled. This allows cross checking of module dependencies and enables an always consistent ECU configuration. Wrong configurations are recognized immediately and several dependencies on ECU level can be resolved automatically. The complete information can be shared with system design tools on one single tool interface or on file level.

- **Extensible:** The eclipse based tool framework is easy extensible to support customer components or legacy components. Also an integration with configurable application software is possible. With the generic configuration and generation application the user can define own modules easily by creating AUTOSAR parameter definition files together with additional GUI description(XML). By using the eclipse environment it is also possible to add custom applications accessing the same data model.

- **Integrated solution:** *tresos ECU* basic software runtime modules are already fully integrated which leads to a significant reduction of effort to be spend in manual integration. With the extensible tool framework also customers modules or legacy components can be integrated in advance.

- **RTE generation:** AUTOSAR compliant RTE generation with optional features for memory protection mechanisms on RTE level and for optimizing resources. If needed the RTE can be generated with high resource optimization by combining the OS and COM configurations and implementations.

- **Debug support:** Enable tracing of different internal informations of the AUTOSAR core. The trace information can be send via existing communication layer (e.g. CAN) or in combination with existing debug interfaces. The configurable trace help to analyze the signal flow through different layers (VFB Trace, tracing of execution time, tracing of protection faults, etc.).

- **AUTOSAR:** Import and export of AUTOSAR basic software component descriptions; import and export of AUTOSAR ECU configurations; import of AUTOSAR system descriptions; import and export of pa-
rameter definition files, import of communication formats (DBC, LDF, Fibex).

AUTOSAR MODELING ELEMENTS

AUTOSAR defines a multiplicity of modeling elements and XML file formats. Those AUTOSAR elements, which are exchanged between system level and ECU centric tools, are explained first to be able to better understand the challenges of an AUTOSAR tool chain.

The system level design tool is responsible for modeling and configuring all aspects which are relevant on system level. Most of these aspects are modeled using elements of the AUTOSAR system description. Nevertheless, the system description also references the software component description which contains all information about software components. This is a list of the most important elements of the AUTOSAR software component description [13]:

- **Software Component Type**: This element describes the properties of one component, e.g. name and ports of the component.

- **Port**: Ports are part of a software component type. A port is a proxy for communication means to external parts such as other components or basic software. Each port is typed by exactly one interface.

- **Interface**: Interfaces describe the elements of components’ ports. AUTOSAR distinguishes between sender-receiver interfaces to describe data flow and client-server interfaces to describe function calls.

- **Composition**: This element describes a hierarchically structured component. A composition itself contains other components. Thus, AUTOSAR software component descriptions are used to specify software architectures, not only for single components, but also for hierarchical components called compositions.

- **Connector**: A connector is used to connect two ports of two components. The interfaces of these two ports have to be compatible. In this context compatible means, that the elements and properties of both interfaces have to be equal.

Figure 4: AUTOSAR Components as illustrated in SystemDesk

Figure 4 shows a composition diagram modeled in SystemDesk. The diagram contains two components, each with one port, connected to each other using a sender-receiver connection. While figure 5 shows a component diagram of one component with a port and interface.

![AUTOSAR Component with Interface](image)

Figure 5: AUTOSAR Component with Interface as illustrated in SystemDesk

An AUTOSAR system description specifies the hardware topology, the network communication and the mapping of software onto hardware:

- **System Topology**: The system topology element describes a system’s topology in the form of ECUs, communication ports and buses with channels and their corresponding attributes like CPU, memory, type of bus, bus speed, etc.

- **System Communication**: This element describes the communication between ECUs in the form of communication matrices. Network messages and signals of each individual matrices are not completely separated, but signals of different matrices can be linked to form exactly one signal on system level. Additionally network messages and signals can contain timing requirements.

- **Top Level Composition**: This is a reference to a composition which is the root software component of the system. The system description itself does neither contain any components nor connectors. It only references a composition which itself is specified using the software component description.

- **Software Component to ECU Mapping**: The components which are modeled on system level are mapped onto ECUs specified as part of the system topology.

- **Data Element to Network Signal Mapping**: Data elements of software components’ ports are mapped onto network signals of communication matrices.

An AUTOSAR system description file exists exactly once per system. In contrast, an ECU configuration file exists per ECU [14]. It contains elements that describe the configuration of basic software of one ECU (operating system, communication stack, I/O, services, etc.). Each file is grouped into different basic software modules. Each module contains a multiplicity of parameters. There are some modules which are directly relevant on system level because they influence the scheduling of runnables and communication between the components:
AUTOSAR-OS: AUTOSAR-OS is an extension to OSEK-OS. Therefore, configuration of OS comprises definitions of tasks, counters, alarms, resources, etc.

AUTOSAR-COM: AUTOSAR-COM is an extension to OSEK-COM. Within the COM configuration, network independent frames and signals along with their properties, like trigger mechanisms and cycle times, are defined.

RTE: The runtime environment (RTE) is responsible of generating production quality C code for communication means between components and to basic software components, and generating tasks which executes the main functions (in AUTOSAR called runnables). Therefore, the configuration comprises selection of communication means (e.g. shared memory vs. AUTOSAR-COM) and runnable to task mapping.

CHALLENGES WITHIN TOOL CHAIN

There arise a lot of open questions concerning the workflow between system level and ECU centric design. This chapter lists different challenges within the workflow. The following chapters propose two favored workflows (top-down and bottom-up) and describe how the challenges can be addressed by an AUTOSAR compliant tool chain.

COMPATIBILITY OF INTERFACES

Ports of components are typed using interfaces as explained in the previous chapter. Two ports can be connected to each other if and only if the corresponding interfaces are compatible. The key features of compatibility are:

- The elements have identical names.
- The data types of the elements are identical.
- The scalings are identical.

Most important is that the scalings of connected data elements have to be identical, so that efficient communication code can be generated automatically.

Ports of application components are connected to each other on the software architecture level as part of the system level. The interfaces are often specified in the very same tool. Thus, interfaces can be easily transferred from one application port to another. If basic software ports are considered the situation is different. On the one hand, the interfaces of the basic software components are specified as part of the basic software configuration and the interfaces of application components are specified on the software architecture level. The challenge persists in finding a workflow to easily specify the interfaces in such a way that they are compatible.

CONNECTING APPLICATION COMPONENTS TO BASIC SOFTWARE COMPONENTS

Application software components are designed independently of the basic software components as explained in the previous section. Nevertheless, at some point in time they have to be connected to each other.

![Figure 6: Explicit Connectors between Application Components and a Basic Software Component](image)

In AUTOSAR there are two ways of specifying these connections:

- **Explicit Connectors:** The user can model explicit connectors (signal lines) between application and basic software components’ ports. The application component and the basic software component are placed hierarchically on the same level. Afterwards, the ports of the components can be interconnected. Figure 6 shows an example with two application components and one NVRAM manager. The ports are connected with explicit connectors. AUTOSAR does not allow explicit connectors between components which do not belong to the same enclosing composition. On the other hand, basic software components are only instantiated once per ECU, i.e. they belong to exactly one composition. They are often placed on the top level composition of the system model. From this it follows that it is difficult to model connections between one basic software component and a set of application components which require the service implemented by the basic software component.

- **Service-Interface:** The user can use identical interfaces at application and basic software ports. That means, that no explicit connectors are modeled, but the same interfaces are assigned to both application and basic software components. These ports are implicitly connected by the generated RTE. The RTE generator searches service interfaces and automatically connects all ports typed with this interface which are mapped to the same ECU.

The challenge persists in finding a workflow and modeling means for specifying both explicit connectors and service interfaces efficiently.
NETWORK COMMUNICATION

The usage of communication matrices is a system level topic because messages have to be used consistently between ECUs. Therefore, an ECU centric tool either has to prevent editing of network communication or has to back propagate the changes to the system level.

TOP-DOWN

The top-down workflow is presented in figure 7. Starting point for this workflow are the application components. Either new components are developed or available components are reused. In both cases, interfaces of application components’ ports are specified in respect of applications’ requirements, and not in respect of basic software components. These interfaces are either specified within a behavior modeling tool or system level design tool. Ports, which will be connected to basic software components, remain unconnected on system level.

At this point, design on system level is finished. Those parts of the system description are extracted which are relevant for each individual ECU. This system extract can be used as a starting point for the ECU centric design.

As explained above, platform independent software architectures specify unconnected basic software ports. During the ECU centric design these ports have to be unconnected to basic software ports. That implies, that the basic software ports are configured in such a way, that they are compatible with the application ports. Thus, the system level design controls the ECU centric design (top-down).

A consistency check concerning the connections between application and basic software components has to be conducted after the basic software has been configured. This comprises:

- Check, whether all basic software ports on software architecture level are connected.
- Check, whether all connected ports are compatible.

Finally, the C code for all basic software modules is generated. It can then be compiled and linked along with the application code.

TOOL CHAIN

The following paragraphs explain how the challenges can be solved within an AUTOSAR compliant tool chain when following the top-down approach.

COMPATIBILITY OF INTERFACES

The interfaces of application components are specified on system level. Nevertheless, these interfaces have to be specified in such a way, that tresos ECU can generate compatible interfaces for the basic software components. Therefore, SystemDesk comprises a library concept. Such a library can contain all standard AUTOSAR service interfaces, like NVRAM, diagnostics or I/O interfaces. These service interfaces can be assigned easily to application components’ ports using drag and drop from the library. The user can even adapt the interface to his needs.

Tresos ECU imports this software architecture comprising the interfaces. The user can configure the basic software components in such a way that compatible interfaces are used for the basic software components’ ports.
CONNECTING APPLICATION COMPONENTS TO BASIC SOFTWARE COMPONENTS

The connection of application components to basic software components is done within tresos ECU. SystemDesk exports a software architecture with unconnected ports. These ports are extracted by tresos ECU so that the user can connect these unconnected ports to the basic software components’ ports which have just been configured.

NETWORK COMMUNICATION

The network communication is fully specified within SystemDesk as part of the system design. This information is represented as communication matrices in the system description XML file. Tresos ECU reads the system description file and automatically generates a consistent presentation of the communication matrices as part of the ECU configuration. These communication matrices are handled read-only to stay consistent with the system level description. Exceptions will be possible for adding debug messages or for testing timing behavior.

BOTTOM-UP

The top-down approach configures the basic software in respect of the requirements of application components. The bottom-up approach does it the other way around. The basic software is configured first and the application ports’ interfaces are specified subsequently. This approach will be used if a hardware already exists and hence basic software parameters are predefined.

As shown in figure 8, the ECU centric tool is used to configure the basic software components of each individual ECU. Basic software component descriptions and the corresponding C code implementation are generated afterwards. These basic software components are added to the system level model. In contrast to the top-down approach, now the system level model contains platform and configuration specific components.

Nevertheless, the interfaces at application components’ ports may have to be adapted to be able to be connected to the ports of the basic software components. I.e. the scaling of data elements may have to be changed to fit the scaling of the basic software. If an adaptation of an application interface is not possible (e.g. for components in object code), a special component for converting the data has to be used.

In this workflow, the consistency check is performed by the system level design tool, i.e. unconnected basic software ports and incompatible interfaces are detected here.

Finally, the runtime environments for all ECUs are automatically generated by either the system level or ECU centric design tool.

TOOL CHAIN

The following paragraphs explain how the challenges can be solved within an AUTOSAR compliant tool chain when following the bottom-up approach.

COMPATIBILITY OF INTERFACES

The ECU centric tool is used to configure basic software components. Afterwards, basic software component descriptions are exported. These descriptions are imported by the system level tool. The basic software components are handled the same way as any other component, e.g. they can be placed into diagrams and ports of basic software components can be connected to application components’ ports. The system level tool than has to check, whether the interfaces of connected ports are compatible. If they are not compatible three possible solutions exist:

- The ECU centric tool can be used to change properties of the basic software components in such a way, that compatible interfaces are generated. This also involves re-generation of the component description.
- The interfaces of the application component are adapted. This often means, that the corresponding C code or the model of the component has to be adapted as well.
- An additional software component is added to the model which converts the data from the input to the output port. We call this kind of component adapter.
CONNECTING APPLICATION COMPONENTS TO BASIC SOFTWARE COMPONENTS

Basic software components are handled as any other component. Therefore, explicit connectors between basic software components’ ports and application components’ ports can be modeled easily using a graphical diagram editor. SystemDesk also supports the concept of service interfaces. Thus, if tresos ECU exports service interfaces for basic software components, these interfaces can be assigned to application components using drag and drop. These ports are than implicitly connected to each other.

NETWORK COMMUNICATION

Network communication is handled in exactly the same manner as for the top-down approach. If e.g. a DBC file is already available, this file can be imported by tresos ECU so that the communication stack can be configured.

PROCESS SUPPORT

As a rule, system level and ECU centric tools are used by different persons, often in different companies. Therefore, both kinds of tools have to be integrated into different software development processes at different companies.

FILE EXCHANGE

A tool coupling on the basis of Windows processes is not suitable because the tools are often run on different PCs. Therefore data should be exchanged on the basis of AUTOSAR compliant XML files. E.g., a software architect specifies a system and generates the corresponding AUTOSAR compliant XML files with SystemDesk. These files can be placed on a file system, sent via email, or even put under version control.

ITERATIVE WORK

The workflows presented in this paper are somehow idealized. Even when using the top-down approach one or more iterations between ECU centric and system level tool might be necessary. This mainly means that update scenarios have to be handled efficiently. E.g. an update mechanism for basic software components is needed for the system level tool. The information regarding this updated component has to be kept whenever possible. E.g., if a basic software component’s port is connected to an application component’s port, and the basic software component is updated, the original connection to the application component has to be kept (as long as the port is still available).

ELEMENT REFERENCING

There might be situations where a problem or inconsistency in the ECU part references elements in the system part, or the other way around. Consider the case, where an application component requires the ECU state manager interface but no ECU state manager is specified. The consistency check of the ECU centric tool has to recognize the unconnected port and report an error which references the application component. This error has to be delegated to the system level tool, so that the user can easily navigate to the unconnected port. The same holds for the other direction between system level and ECU centric tool.

CONCLUSION

AUTOSAR specifies file formats for the development of series ECUs’ software. It also defines a waterfall-like methodology. Iterative workflows between tools are only considered marginally. This is appropriate from the AUTOSAR perspective. Nevertheless, tools must be flexible in respect to the customers processes.

Within a cooperation, dSPACE and Elektrobit worked out the requirements for an integrated AUTOSAR tool chain and incorporated these requirements into dSPACE’s new system level design tool SystemDesk and Elektrobit’s ECU centric tool tresos ECU. Besides the technical features presented in this paper the cooperation between dSPACE and Elektrobit also brings benefits in other respects: The progression of the AUTOSAR specification is an ongoing process. Thus, the specifications are incomplete and partly imprecise. The coupling of SystemDesk and tresos ECU is tested intensively. This also includes performance issues when using realistic AUTOSAR models. The test stage is not yet finalized. Therefore quantitative results are not yet available.

Open issues in respect to the AUTOSAR specification are detected and solved consistently within the cooperation. The open issues and their solutions are also fed back into the working groups of the AUTOSAR consortium. Last but not least the roadmaps of SystemDesk and tresos ECU are agreed so that consistent versions of the tools are available.

In combination with dSPACE’s production quality code generator TargetLink, dSPACE and Elektrobit offer an integrated tool chain for the model based software development of series production ECUs and networks of ECUs based on the AUTOSAR methodology.

REFERENCES

[1] AUTOSAR. Homepage at www.autosar.org


