MASTER

Methodology for selecting low-cost AUTOSAR tool-chain and its evaluation through a case study

Janardhan, J.

Award date:
2018

Methodology for selecting low-cost AUTOSAR tool-chain and its evaluation through a case study

EIT Digital Embedded Systems - Master Thesis

Janahvi Janardhan
j.janardhan@student.tue.nl

Systems Architecture and Networking Group
Department of Mathematics and Computer Science

Eindhoven University of Technology

Supervisor:

Dr. ir. Reinder J Bril, TU/e

Committee Members:

Dr. ir. Ion Barosan, TU/e
Diederik van Dijk, BRACE Automotive
Geert van der Wal, BRACE Automotive

Version 1.0

Eindhoven, August 2018

Abstract

Automotive embedded systems are going through a rapid paradigm shift in terms of embedded system architectures and software design techniques. The increasing complexities have led to a shift from using legacy software towards using the AUTomotive Open System Architecture (AUTOSAR). AUTOSAR is an open and standardized software architecture for the development of automotive Electronic Control Units (ECUs), jointly initiated by manufacturers, suppliers and developers. AUTOSAR tools to implement the AUTOSAR architecture have found a great prominence due to the increasing complexity of this standard and hence they are highly priced. This thesis explores a methodology for selecting AUTOSAR tools by the application of Analytic Hierarchy Process (AHP). Each tool alternative is ranked based on six main selection criteria viz., Functionality, Interoperabiltiy, Usability, Service & Support, Cost & Distribution and Testability. Further, an emphasis is given to selecting low-cost AUTOSAR tools, exploring the opportunities for new entrant automotive companies venturing into the AUTOSAR market. Methodology applied for selecting tools in this thesis led to following result: Performance wise AUTOSAR tools from Vector, Dassault Systems and ETAS were ranked high. Cost wise, ArcCore's Arctic Studio tool, COMASSO and Mathwork's Embedded Coder got higher ranks. Out of these three, Arctic Studio was used for implementing an AUTOSAR compliant Controller Area Network (CAN) communication stack. It was further evaluated based on the selection criteria mentioned above. Based on the experience gained, it could be concluded that, it is crucial that a tool vendor provides support for the hardware platform selected for development of an AUTOSAR application.

Acknowledgments

I would like to express my sincere gratitude to my committee chair, Dr. ir. Reinder J Bril and Dr. ir. Ion Barosan for their immense support, guidance and patience during this entire thesis. I sincerely thank Mr. Bart Oosthoek, Mr. Diederik van Dijk, Mr. Geert van der Wal and Mr. Ruud Bogers of BRACE Automotive for the opportunity, their feedback, support and advise. I would like to also thank Mr. Daniel Versteeg, Mr. Marco Stijn and Mr. Onno Oenema of Orlaco B.V. for giving me an opportunity to collaborate with them during this thesis. In addition, a thank you to Mr. Siddharth Nair and Mr. Thomas Winkeler of ArcCore for their gracious support with Arctic Studio. Last but not the least, I'd like to thank all my colleagues at BRACE Automotive, family and friends for standing beside me during difficult times.

Contents

List of Figures

List of Tables

Listings

Abbreviations

ADC	Analog to Digital Converter
AHP	Analytic Hierarchy Process
ALMA	Architecture Level Modifiability Analysis
ANP	Analytic Network Process
API	Application Program Interface
ARXML	AUTOSAR Extensible Markup Language
ASW	Application Software
AUTOSAR	AUTomotive Open System Architecture
BSW	Basic Software
BswM	Basic Software Manager
CAN	Controller Area Network
CANIF	Controller Area Network Interface
CCS	Code Composer Studio
CDD	Complex Device Driver
CI	Consistency Index
COM	COMmunication
COMET	Characteristic Object METhod
COMM	COMmunication Manager
CR	Consistency Ratio
DIO	Digital Input Output
E/E	Electric / Electronic
ECU	Electronic Control Unit
ECUAL	ECU Abstraction Layer
EcuC	ECU Configuration
EcuM	ECU Manager
EEPROM	Electrically Erasable Programmable Read-Only Memory
ETHIF	ETHernet Interface
IoHwAb	IO Hardware Abstraction
IPdu	Information Protocol Data Unit
MCAL	Microcontroller Abstraction Layer
MCU	Microcontroller Unit
MEMIF	MEMory InterFace
OEM	Original Equipment Manufacturer
OS	Operating System
PduR	Protocol Data Unit Router
PWM	Pulse Width Modulation
RI	Random Index
RTE	Run-Time Environment
SAAM	Software Architecture Analysis Method
SL	Service Layer

SMAA	Stochastic Multicriteria Acceptability Analysis
SPI	Serial Peripheral Interface
SysML	System Modeling Language
VFB	Virtual Function Bus

Chapter 1

Introduction

1.1 Background

Over the past few years, the automotive industry is experiencing a new revolution and is adapting to new challenges with developments in technology, consumer demands, globalization and collaboration strategies. The Electrical and Electronic (E/E) complexity and the amount of software code within a vehicle has increased tremendously and it continues to grow [1]. The innovative functionalities and software features incorporated within modern cars are a key differentiator between a high-end and a low-end vehicle [2]. About 80% of the innovative applications in the vehicles today, like the adaptive cruise control, collision avoidance, parking aid, in-car entertainment, etc., are all based on a collection of embedded software and hardware features [3] (Figure 1.1). With the increase in the amount of these applications, the number of Electronic Control Units (ECUs) deployed in a car, has also increased exponentially [4]. This leads to an overall increase in the hardware, software and network complexities [5]. As a result of increasing complexity of ECUs in vehicles, an automotive engineer is faced with numerous design challenges across a wide range of applications.

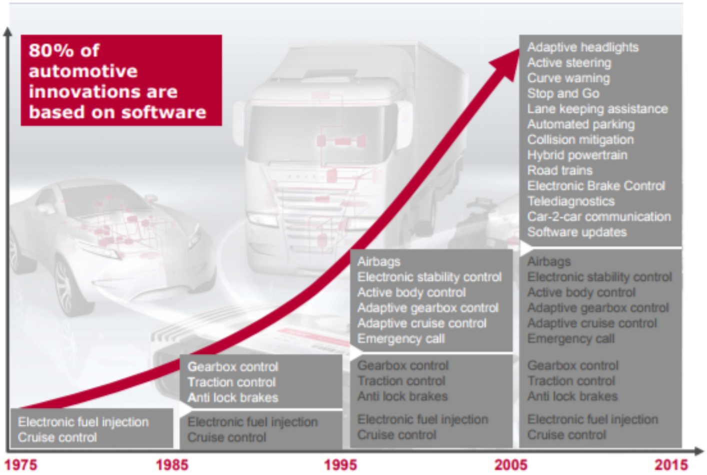


Figure 1.1: Exponential growth of ECU complexity

In a traditional method of designing ECU software architecture, the Original Equipment Manufacturers (OEMs) and the software developers followed an ECU-centric development approach. In an ECU-centric design approach, adding new software components to the existing software required the redesign of an entire ECU software architecture [6], which hindered non-functional demands like software re-usability and shorter time to market. Moreover, the last generation cars did not have many applications that were computationally intensive. With the onset of autonomous cars and advanced applications deployed on it, the demand for a high performance hardware has increased. Factors like lack of scalability, reusability and interoperability across product lines; need for more flexible solutions and reduced complexity in software development led to a paradigm shift from ECU-centric approach to focusing on function-centric development. This motivated the creation of the AUTomotive Open System ARchitecture (AUTOSAR) consortium which was formed by major automotive OEMs like BMW, Ford, Daimler Chrysler etc. in the year 2003 [8] [9].

AUTOSAR is an open standard which aims to standardize the automotive software architecture and framework. It is a component based reference architecture for automotive software applications which also deploys a layered architecture style for developing software layers for automotive ECUs. The layered architecture decouples the above layer from the layers below, thereby providing abstraction and masking the underlying details. This separates the concerns for software developers, system designers, system integrators etc. The AUTOSAR architecture, decouples the Application Software (ASW) layer from the underlying Basic Software (BSW) layer by means of a standardized middelware called the Run-Time Environment (RTE) layer. The standard also enables non-functional requirements like scalability, transferability, interoperability, to name a few. Transferability refers to reducing the overhead involved in transferring functions between ECUs and different platforms, which implies an increased code re-use. Scalability refers to a possibility to add and remove functions without having to re-configure the underlying code mapped to the hardware. Interoperability means the ability to integrate the functional modules from multiple suppliers [7]. It's also aimed at enhancing the quality and reliability of the E/E systems. The main goal of the AUTOSAR consortium is to agree upon co-operating on the standards but to compete on implementation.

AUTOSAR partnership has varied levels of membership [14], Core members, Premium members, Development partners, Associate partners and Attendees. The core partners are involved in the development and management of the AUTOSAR standard and specifications. The premium members have to contribute 1.5 of a full-time equivalent (FTE) and also contribute €17,500 annually, towards the consortium. The development members are required to contribute 0.5 of an FTE, while the Associate partners are required to contribute €10,000. However, the attendees do not have to contribute either in the development time (FTE) or contribute any annual fee. This means, they are the only group that are not allowed to use AUTOSAR royalty-free for developing AUTOSAR applications [8]. Therefore, in order to use AUTOSAR products commercially, an organization must be a member of the AUTOSAR partnership.

Although, AUTOSAR architecture was developed to ease the process of software development of E/E systems, constant addition of new features and extensions to the standard has made the standard itself more complex. This implies that, implementing AUTOSAR standard is a complex task to manage via manual work-flow and requires a sophisticated tool-chain for functions like modeling, early stage development, verification, validation and testing of the system. Tools are used in the development of the AUTOSAR architecture right from model design phase to configuring the application components, basic software components, run time environment and finally generating executable files (final product). Furthermore, these tools play an important role in improving lead time, time to market and deliver cost advantages for the OEMs by enabling the use of standard interfaces and components based on the AUTOSAR specifications. As a result, there is a huge demand for using an AUTOSAR tool-chain by major OEMs and tier-1 suppliers in automotive industry.

In order to deploy the AUTOSAR architecture on automotive ECUs, the AUTOSAR tool-chain plays a pivoting role. AUTOSAR standard specifications are very extensive and time consuming to be implemented manually (> 20,000 pages) [15] and keeping up with lead time for product development is of utmost importance in the automotive industry. Therefore, selecting a right tool-chain forms an integral part of the development process of an AUTOSAR compliant automotive ECU.

This thesis emphasizes on an increased importance of selecting and using an appropriate tool-chain to implement AUTOSAR architecture and automate the software development process.

1.2 Problem Description

In the past few years, many tool vendors have incorporated AUTOSAR tooling into their product list. It was observed that there were mainly two categories of tool vendors that developed these tools. Some tool vendors enhanced their existing tooling to include certain export and import capabilities of AUTOSAR features within their tools (e.g. Continental). Others used the standard specifications as an opportunity to create new AUTOSAR tools (e.g. Vector, ArcCore). These tools either provided functionality covering only a part of the entire design and development phase (i.e. modeling, deploying, testing), or provided functionality that covered all development phases. With different combinations of these tools it is inconvenient for OEMs to find the most appropriate set of tools that support their particular E/E architecture design process. Even with a proper set of tools, it remains a major challenge to combine them into a consistent and flexible tool-chain, covering all design and developmental phases. In particular, there is a lack of a systematic approach (methodology) towards selecting these tools.

Further, selection of requirements for developing tools by prominent tool vendors [11, 12] often benefits the OEMs and other multi-national tier 1 companies. The tools are made to be highly sophisticated to handle the growing requirements on E/E architectures and ease the process of production, thereby increasing the productivity and profit for the OEMs. As a result, the majority of tools are proprietary and high-priced. Those companies which are below the tier 1 zone (small OEMs, automotive equipment manufacturers, new entrant companies), find *"initial investmest"* to be one of the major prohibitive factors to procure an appropriate set of tool-chain [10]. For example, Vector [11], one of the tier 1 companies [14] and one of the major competitors in the AUTOSAR tool-chain market, provides a set of tools for implementing AUTOSAR architecture on automotive development platforms. Although these tools are full-fledged and sophisticated, they are very expensive to procure for a new entrant AUTOSAR company in the automotive market [10].

In order to reduce the initial expenditure, it is important to understand what factors affect the performance and increased cost of the AUTOSAR tool-chain. There are many aspects that must be considered before making such a huge investment. Though AUTOSAR standard specifications provide the necessary requirements for tools [17], tool vendors add special features to increase their competitiveness. While this is not bad, it might not be a feasible low-cost option. Lack of a methodology to select an AUTOSAR tool-chain and in particular low-cost tools is the main motivation for this thesis. The purpose of this thesis is to provide a methodology for selecting an AUTOSAR tool-chains and in particular, prioritizing cost as the main selection criteria to select a set of tool-chain to implement the AUTOSAR architecture.

This thesis was conducted at BRACE Automotive B.V., who are a technological solutions provider in the automotive sector, located in Eindhoven, the Netherlands, partnered with Orlaco B.V., who are specialized in providing camera solutions, located in Barneveld, the Netherlands. BRACE Automotive came up with an idea of finding low-cost tooling options w.r.t. AUTOSAR and to further understand the AUTOSAR architecture and how it works. Both BRACE Automotive and Orlaco were interested in understanding the required initial investment for using AUTOSAR and gaining the AUTOSAR knowledge and trade-offs involved in selecting low-cost tool-chain. To support the investigation, a case study was performed, that resulted in a demonstrator that implements a CAN communication stack using a low-cost AUTOSAR tool.

For selecting an appropriate tool-chain, a list of criteria had to be considered and weighted, upon which the tools could be compared. Therefore, a list of Multi-Criteria Decision Making (MCDM) methods were investigated in order to select an AUTOSAR tool-chain for a particular criteria, which is explained in Chapter 2. The selected MCDM algorithm is further evaluated to judge its performance. The demonstrator, on the other hand, is used to evaluate the selected low-cost AUTOSAR tool. In the remainder of this chapter, research objectives and research questions are addressed together with the research methodology and is concluded with an outline for the rest of the report.

1.3 Objectives

In order to realize the main goal i.e. to develop a methodology to select an AUTOSAR tool-chain having cost as a key criteria, the following research objectives have been stated. (Here the system under consideration is the AUTOSAR tool-chain)

1. *Identification of the available MCDM tools and selecting the one that works best for this application (to select AUTOSAR tool-chain).*
2. *Identification of stakeholders who work with the AUTOSAR tool-chain and discovery of the key requirements for each of these stakeholders. This also includes identifying the key constraints and other bottlenecks involved in requirement analysis.*
3. *Identification of the key criteria for developing a methodology for selecting an AUTOSAR tool-chain (which included basic components and/or framework for a specific class of products), with the help of a decision making tool. The tool-chain selected was used to develop the demonstrator.*
4. *Implementation of the demonstrator (artifacts of the demonstrator), evaluation of the tool-chain used to implement the demonstrator. Further, also evaluating the performance of MCDM tool used in selecting the AUTOSAR tool-chain.*

1.4 Research Questions

To achieve the project objectives listed in Section 1.3, the following research questions were formulated:

- Initially, a research was done on the available Multi-Criteria Decision Making (MCDM) tools. Required knowledge for each of these methods was gained. Now, it was important to decide which method to use and why. This led to the first research question which subsequently answers the first objective.
  RQ1.1 *Which MCDM method is likely to perform better for this application and why?*
- After gaining the required theoretical knowledge about AUTOSAR, key stakeholders involved in the process of developing an automotive ECU and their key concerns were identified. Normally, a stakeholder is typically involved with the development of a system (demonstrator) as an end product. But in this context, the goal was to select a tool-chain according to a certain criteria. As a result, stakeholders considered are typically the ones that make use of the AUTOSAR tool-chain and requirements are gathered based on what tool features are important for developing an automotive ECU seamlessly. Hence, "AUTOSAR tool-chain" is regarded as "system" in this context. Next set of research questions are formulated accordingly (second objective).
  RQ2.1 *Who are the key stakeholders in development of an automotive ECU using AUTOSAR architecture?*
  RQ2.2 *What are their key concerns, requirements and constraints in terms of using AUTOSAR tool-chain to develop automotive ECUs seamlessly?*

- After having gathered all the stakeholders requirements and fine tuned, next step was to analyze and come up with key criteria in selecting the tool-chain. To analyze these requirements, architectural diagrams and models using SysML were considered. This led to the next set of research questions (second objective).
  RQ3.1 *What are the different architectural views and models of the AUTOSAR tool-chain that addresses the concerns of stakeholders?*
  RQ3.2 *How to derive the key criteria for selecting the tool-chain from these models?*
  RQ3.3 *What are the different criteria that are finally selected in order to opt for an AUTOSAR tool-chain?*

- After having found what are the key criteria and having made a list of tool vendors available in the AUTOSAR tool market, the next step was to quantify the criteria. For this purpose, a Multi-Criteria Decision Making tool was applied. Two sets of results were gathered from this approach. One, to come up with the tool that outperformed others in terms of performance and hence a big OEM looking for selecting AUTOSAR tool-chain purely based on performance could opt for this tool. But if a small company that has a certain restriction on the money that can be spent, then the other result depicted the set of tool-chain with cost as the key selection criteria over others. Accordingly, these were the third set of research questions (third objective).
  RQ4.1 *Which tool outperforms others in terms of performance as the key selection criteria?*
  RQ4.2 *Which tool is better when cost is considered as a key criteria for selecting AUTOSAR tool-chain?*
  RQ4.3 *What are the trade-offs made in answering the question RQ4.2?*

- To achieve the final objective, a demonstrator was implemented which is explained in Chapter 5. This step was the most crucial step to evaluate the tool-chain selected. Further, the decision making method used was also evaluated based on the experience gained. The following set of research questions were answered in this regard (fourth objective).
  RQ5.1 *Did the selected low-cost tool-chain perform well when applied to the demonstrator?*
  RQ5.2 *What were the various challenges faced in this regard?*
  RQ5.3 *How did the selected MCDM tool perform? Are there any recommendations to improve the selection criteria or methodology to select the AUTOSAR tool-chain?*

1.5 Research Methodology

Research methodology describes the procedure adopted in this work to answer the research questions in Section 1.5 and to achieve the set of goals and objectives for this thesis.

1. At first, a literature analysis of the available MCDM tools was done. After having understood the pros and cons of each method **(RQ1.1)**, a suitable method / decision making tool was selected.

2. Next, a basic knowledge about AUTOSAR architecture standard, methodology and tools required was acquired by studying the AUTOSAR standard specifications, journals and other web resources. Then, stakeholders (that develop automotive ECU) who use the AUTOSAR tool-chain were identified and requirements and concerns were put forth **(RQ2.1) (RQ2.2)**. Accordingly, constraints in these requirements were identified.

3. Having understood the requirements of the stakeholders, the architectural description and viewpoints were modeled with the help of architectural diagrams **(RQ3.1)**. The results obtained in this step helped in understanding how each stakeholder interacts with different parts of the tool-chain and this eventually helped in identifying the key criteria required to select an AUTOSAR tool from the list of tool vendors considered **(RQ3.2 RQ3.3)**.

4. The selected criteria are further quantified using the MCDM tool and the AUTOSAR tools which outperform others in terms of performance and cost are then selected **(RQ4.1, RQ4.2)**. In order to select cost over performance, some trade-offs were made **(RQ4.3)**.

5. The selected low-cost tool-chain was then applied to build the demonstrator, and it was further evaluated using cost-to-performance ratio **(RQ5.1)**. Final conclusions were made by documenting the challenges faced and recommendations to improve the selection criteria or the methodology**(RQ5.2, RQ5.3)**.

1.6 Thesis Outline

This section explains the structure of the report. Chapter 2 imparts knowledge on the AUTOSAR architecture briefly. Chapter 3 presents a list of decision making methods and analysis. Chapter 4 explains the methodology for selecting a tool-chain by considering stakeholders and their requirements. Chapter 5 describes the system design for the demonstrator. Chapter 6 explains the implementation and testing process in detail. Chapter 7 focuses on evaluating the tool-chain that was used in the implementation of the demonstrator and the evaluation of MCDM tool chosen. Chapter 8 presents the challenges faced, lessons learned and conclusions along with future scope.

Chapter 2

Decision Analysis Methods

Section 2.1 briefly describes prior work done with respect to model-based design tools in automotive industry and the transition to AUTOSAR architecture and tools (Chapter 3), which further motivates the need for adopting decision making algorithms. These algorithms are used for applications in various fields like research and development, management, strategic planning and others. After analyzing all the Multi-Criteria Decision Making (MCDM) algorithms, the Analytic Hierarchy Process (AHP) was selected. The working of AHP and its application in order to select the AUTOSAR tool-chain is described in Chapter 4.

2.1 Model-based Software Development tools

With an increase in the number of ECUs in a modern car, software complexity has also increased over time. In order to address the growing complexity of software applications and algorithms, automotive engineers have incorporated Model-Based Design (MBD) approach in developing ECUs. Model-based design, which is a widely used and accepted approach, is a methodology applied in designing embedded software systems. It's a mathematical and visual method of addressing problems associated with designing complex control, signal processing and communication systems. It imparts automatic verification and validation, dynamic hardware-in-loop simulation, code generation and many such other benefits, which enables the developer to identify errors at a very early development stage.

MBD is widely used in the automotive and avionics domain today and hence considered as one of the best approaches to handle complexities of modern embedded systems which are real-time and safety critical [19]. This is because, MBD and software development together with tool integration are getting more agile in development process. This improves an overall consistency of the system. However, Broy et al., also pointed out that, though MBD is widely accepted, most engineers still make use of standalone tools and therefore adapt their engineering methods and processes to available / legacy tools. Moreover, integration of a tool-chain is not seamless and to overcome the challenges of tool integration [21] a deep, coherent and comprehensive integration of models and tools were required.

Further, Holtmann et al. [20], mentions about the process and tooling gaps present between different modeling aspects for the system being developed. The proposed tool-chain in this paper mentions two important tooling gaps. One, missing links between system level tools and software development tools. Two, tools that are not inter-operable and require manual synchronization and hence often inconsistent (rely on redundant information) and due to lack of automation require redundant manual work.

Here, in order to mitigate the missing links between system architecture and software architecture, Boldt [22] mentions that using Unified Modeling Language (UML) and System Modeling Language (SysML) is the most powerful and extensible way to bridge the gap. Traceability of requirements and different models can be achieved using UML and SysML, according to the author. Pagel et al. [23] describes the benefits of using XML schema for data exchange via different tools.

The inception of AUTOSAR architecture along with SysML/UML was intended to bridge this gap between system architecture description and software architecture description further. Broy et al. [19] describes how AUTOSAR is one of the major approaches to create an integrated product model for automotive domain. Although AUTOSAR provides a standardized approach for software architecture description and data transfer (XML language), the authors also point out that, due to a missing common tooling platform, the resulting AUTOSAR tools from different tool vendors are again not fully compatible.

This led to an automotive tool-chain for AUTOSAR as presented by Voget [24] called ARTOP which provides a common base functionality for development of AUTOSAR compliant tools. ARTOP is built on the Eclipse platform serving only as a common base for AUTOSAR tool development and is not a tool solution in itself [25]. The available AUTOSAR tool vendors adopting ARTOP in their developmental platforms are *expected* to provide interoperability and easy integration among tool-chains. Further, there are also other important criteria that has to be considered other than just the two mentioned above. As a result, in order to select an appropriate set of AUTOSAR tool-chain, determining a right set of criteria and ranking the alternatives required a more systematic approach. Therefore as a point of reference for this work, decision making tools were considered, which is further explained in next section.

2.2 Decision Making Tools

Decision making is an effective way of choosing between two or more alternatives for a particular task. For example, a System Architect of a certain company has to make decisions each day. Though management takes the final decision, the System Architect is responsible to give crucial insights and necessary data required to make a correct analysis and arrive at a decision. According to Hammond et al. [27], a decision making process involves a clear understanding of problem, goals and objectives. Then, the required information is gathered to support the decision and different alternatives are identified. If there are multiple conflicting criteria in making such decisions like cost, quality, usability, interoperability etc., then using a decision making tool takes into consideration all the criteria and helps in choosing a best alternative.

It is observed that there is no single MCDM method to meet the requirements of every application. A most appropriate decision making method for one application may not be a perfect fit for another application. A feasible way to select an appropriate decision making method is by assessing the attributes of an application to which it is being applied to, assessing the characteristics exhibited by various MCDM methods under consideration and finally select an appropriate method.

In this case, the main purpose was to select an appropriate MCDM method which in turn helps in ranking tool alternatives for implementing the AUTOSAR architecture. Accordingly, following objectives were considered before selecting a suitable MCDM method.

1. The selected MCDM method should be able to decouple goals from criteria and alternatives in order to make the *AUTOSAR tool selection methodology* applicable to all instances. For example, the selected MCDM method must allow adding or deleting alternatives and criteria with little or no modifications to other values. Therefore, a distinct hierarchy makes it possible to handle all dimensions of the selection criteria dexterously.

2. The selected MCDM method should be able to consolidate weighted criteria for each goal.

Since implementing an AUTOSAR architecture is an aggregation of more than one AUTOSAR tool, a cumulative comparison has to be made between the alternatives for a particular criteria. As a result, the selected MCDM method should be able to rank the alternatives based on consolidated weighted criteria. The selected MCDM method should be able to take uncertainty factors during comparison into account.

3. The total number of tool alternatives being considered for comparison in order to select an appropriate AUTOSAR tool-chain is not more than 15 tool vendors. It is known that as the number of alternatives increases, the number of comparisons to be made also increases. As a result, the selected MCDM method should be feasible to handle upto 15 alternatives and finish in finite amount of time.

4. The selected MCDM method must be feasible and accessible to be performed in a finite amount of time. Further, in order to reduce the decision's makers workload, the selected MCDM method must be able to completely or partially automate some operations.

Based on above mentioned objectives, the most important criteria considered to select a suitable MCDM method are as follows:

- Distinct hierarchy
- Effective numerical analysis and core concept of the method
- Size of the application
- Feasibility measure
- Automation

Next section presents a list of suitable MCDM methods and are explained in brief. Further, all these methods are compared based on the selection criteria mentioned above. Finally, an appropriate MCDM method is selected which is further applied to select a suitable AUTOSAR tool-chain in Chapter 4.

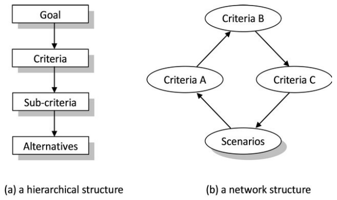
2.2.1 Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process [29], was developed by Thomas L. Saaty in the 1970s as a tool to allocate resources and planning needs for the military. However, due to its ability to prioritize multi-criteria requirements efficiently, it has been adopted in various fields like business, government, industry, education etc. It is considered to be a structured technique for organizing and analyzing complex decisions based on mathematics. AHP follows a mathematical approach of allocating weights to each criteria and evaluating each alternative before arriving at a decision. AHP tool is usually applied for criteria and alternatives that are mutually exclusive from each other and the number of alternatives are not too many. As a result, AHP method was considered to be a close fit for this case.

2.2.2 Analytic Network Process (ANP)

The Analytic Network Process [31], is a more general form of the Analytic Hierarchy Process used in multi-criteria decision making analysis. AHP structures a decision problem into a hierarchy with a goal, decision criteria, and alternatives, while the ANP structures it as a network. That means, AHP considers each element in the hierarchy (criteria and alternatives) to be independent of each other, while ANP takes into account the interdependence within the hierarchy.

In this case, since each alternative (AUTOSAR tools) and criteria were exclusive, ANP algorithm was not an accurate choice. Exclusivity of criteria and alternatives is because, the AUTOSAR standard specifies that all tool vendors should adhere to the standard data exchange formats while sharing information, thus making them interoperable [17].



(a) a hierarchical structure

(b) a network structure

Figure 2.1: Exponential growth of ECU complexity

2.2.3 Data Envelopment Analysis (DEA)

Data Envelopment Analysis (DEA) is a linear programming based technique for measuring relative performance of organizational units, which was first developed by Charnes, Cooper and Rhodes in 1978 [36]. It's mainly used as a benchmarking technique to improve organizational performance by comparing resources used and services provided by the organization. Further, it also helps in identifying the most efficient units or best practice units and the inefficient units where efficiency improvements are possible. This method was not considered in this case as we are only interested in ranking and selecting the best tool-chain that are already available in the market.

2.2.4 ELimination Et Choix Traduisant la REalité (ELECTRE)

ELimination Et Choix Traduisant la REalité (ELECTRE) in English means Elimination and Choice Translating Reality, is a well-known MDCM method particularly in Europe. ELECTRE group of methods was proposed by Bernard Roy of France in 1960s [32]. It is a multi-criteria decision analysis based on outranking method, consisting of several different models (I, II, III, IV, A, IS and TRI) which are derived from the original ELECTRE I.

Outranking method based on ELECTRE amounts to validating or invalidating a pair of alternatives. ELECTRE's preference based relations are modeled via a system of binary outranking relations for each criteria separately. Construction of outranking relations are based on Concordance and Non-discordance concepts. With these concepts, ELECTRE methods build one or several outranking relations i.e., crispy, fuzzy, embedded. Considering two alternatives 'a' and 'b', in a crispy outranking relation 'a' is *strictly preferred* to 'b' or vice versa. In a fuzzy relation, 'a' is *indifferent* to 'b' and in an embedded relation 'a' is *incomparable* to 'b'. Further, an exploitation procedure is used to exploit previously obtained results and obtain the desired results for a given problem.

This method was not considered suitable in this case since a single AUTOSAR tool doesn't provide all the required features and it is clear that a tool-chain from more than one tool alternatives must be considered for ranking.

2.2.5 Fuzzy Set Theory

Fuzzy set theory is an extension of classical set theory where elements have a degree of membership compared to the boolean logic of classical sets. It was first proposed by Lofti Zadeh and Dieter Klauma in 1965 [37]. The decision making using fuzzy set theory method is mainly used in fields like artificial intelligence, traffic monitoring system, human speech recognition, weather forecasting systems etc. Although, fuzzy set theory has wide range of applications, in this case it was not suitable to apply.

2.2.6 Goal Programming (GP)

The term Goal Programming (GP) was first coined in a book published by Charnes and Cooper in 1961 [38]. GP is a multi-objective programming technique. It can be thought of as an extension of linear programming to solve complex decision variable problems where several objectives as well as many variables and constants are involved. The basic approach of GP is to establish a specific numeric goal for each of the objectives, formulate an objective function for each objective and then seek a solution that minimizes the weighted sum of deviations of these objective functions from their respective goals. This approach was not suitable for this case since there was a single and a clear objective set to achieve.

2.2.7 Multi Attribute Utility Theory (MAUT)

Multi Attribute Utility Theory [28] is yet another decision making tool used to make analysis for a decision problem that focuses on the structure of multi-criteria or multi-attribute alternatives usually in the presence of risk or uncertainty. It can also be applied on methodologies for assessing individual's values and subjective probabilities. It is also based on mathematical theory and the information obtained from the assessment usually is fed into the parent problem to rank alternatives and make a suitable choice. Sensitivity analysis is also involved in the assessment and choice processes. AHP algorithm is a direct competitor to MAUT as an efficient MCDM technique for ranking the alternatives. However, MAUT is suitably used for solving problems with a large number of criteria and alternatives, which are subject to constraints. Since the number of alternatives chosen for this case were not too large, AHP was chosen over MAUT.

2.2.8 Preference Ranking Organization METHod for Enrichment Evaluation (PROMETHEE)

PROMETHEE method, like ELECTRE, is yet another family of outranking methods which was first introduced by Jean-Pierre Brans in 1982 [33]. It includes PROMETHEE models I, II, III, IV, V, VI, GDSS, TRI and CLUSTER. Partial and complete ranking of alternatives with the methods of PROMETHEE family is made by calculating a positive outranking flow and a negative outranking flow for each alternative. This method was not considered for the same reason as ELECTRE.

2.2.9 Simple Addition Weighting (SAW)

Simple Addition Weighting (SAW) [34] is based on a value function which simply performs addition of scores that represent the goal achievement under each criterion and are multiplied by particular weights. It is mainly used in applications like water management, business, financial management etc. Though it is very simple to use it is not very popular among decision makers.

2.2.10 Simple Multi-Attribute Rating Technique (SMART)

SMART is the simplest version of MAUT which was first introduced by Edwards in 1971. SMART technique is based on linear additive model where the ratings for alternatives are assigned directly instead of pair-wise comparison. For example, the criterion "top speed" for cars would range from 150 to 200 miles per hour. Thus alternatives that satisfy this criteria are selected. Since each AUTOSAR tool differs qualitatively and it's not feasible to rate the selection criteria directly, SMART method is not chosen for this case.

2.3 Summarizing the comparison of MCDM methods for this application

Table 2.1: AUTOSAR tool features coverage

MCDM algorithms	Project Size	Distinct Hierarchy	Effective numerical analysis and core concept	Feasibility measure	Automation	Selected? / Not Selected?
AHP	Suitable for applications with less than 15 alternatives	Yes	Pair-wise comparisons	Feasible (Expert Choice, Excel etc.)	Yes (by specifying suitable thresholds for automatically deciding some pair-wise comparisons)	Selected
ANP	< 15 tool alternatives	No. Suitable for applications with hierarchical dependencies	Pair-wise comparisons	Feasible (Excel)	Yes	Not selected because of interdependence between the hierarchies
DEA	Can handle large number of inputs	No	Linear programming, measuring relative performance	Yes (mainly used by large organizations to increase their productivity)	Yes (use of macros to automate the process of calculating the efficiency of each unit)	Not selected, since this method was not suitable for this application
ELECTRE	< 15 alternatives	Yes	Outranking method	Feasible (Excel)	Yes (by providing thresholds and arriving at concordance to each requirement)	Not selected because outranking method was not suitable for this application

Fuzzy	Can handle large number of alternatives	No	Comparison of elements based on a degree of membership to a particular set	Feasible (Excel)	Yes	Not selected since this method is not suitable for this application
GP	Can handle large number of inputs	No	Multi-objective programming technique	Feasible (Excel)	Yes	Not selected since this application has a clear objective
MAUT	Can handle large number of input elements	Yes	Ranking of multi-criteria or multi-attributed alternatives	Feasible (Excel)	Yes	Not selected because the number of alternatives considered for this application was not too large
PROMETHEE	< 15 alternatives	Yes	Outranking method	Feasible (Excel)	Yes (automated selection of certain parameters like criteria weights, preference function thresholds)	Not selected because outranking method was not suitable for this application
SAW	< 10 alternatives	No	Additive model	Feasible (Excel)	Yes	Not selected
SMART	Can be used for large number of input elements	Yes	Additive model	Feasible (Excel)	Yes	Not selected because additive model is not suitable for this application

Thus, the research question **RQ1.1** is answered.

Chapter 3

Literature on AUTOSAR Architecture and Tools

This chapter provides a brief overview of the AUTOSAR architecture and the required AUTOSAR knowledge. It is mainly divided into three sections, Section 3.1 to 3.3 provides a brief overview on all the three main aspects of the AUTOSAR architecture i.e. software architecture, methodology and interfaces. Section 3.4 provides a detailed description of the BSW modules that are further required. Section 3.5 provides more information about the AUTOSAR tools and current scenario in the AUTOSAR tool market.

3.1 AUTOSAR Layered Software Architecture

The layered architecture style [40] organizes a system into a set of layers each of which provides a set of services to the layer above. The decoupling mechanism of the layers above from the layers below hides the unnecessary details and minimizes the complexities of each layer thereby imparting abstraction and encapsulation properties.

In the AUTOSAR architecture, adopting the layered architecture style, enables decoupling application development process from the underlying hardware. This property allows a software developer to develop an application for an ECU without being concerned about the hardware architecture, hence making the whole process function-centric rather than ECU-centric. This approach also enhances software re-usability for OEMs because of the standardization of software modules. Figure 3.1 shows the skeletal framework of the AUTOSAR architecture which mainly includes application layer, Run-Time Environment (RTE) layer, Basic Software (BSW) layer, all built on top of the underlying micro-controller hardware.

The application layer mainly consists of software components and BSW layer consists of system software modules as shown in Figure 3.2. There are two ways to classify these modules, viz., vertical and horizontal. In vertical classification, BSW modules are classified as system stack, memory stack, communication stack, I/O stack and complex drivers. Horizontal classification (which is color coded) includes Services Layer (SL) (modules in purple), ECU Abstraction Layer (ECUAL) (modules in green) and Micro-controller Abstraction Layer (MCAL) (modules in red). RTE layer conjoins the application layer and the BSW layer and enables communication. Micro-controller is the hardware board on top of which the AUTOSAR architecture layers are implemented.

3.1.1 Application Layer

Application software component is an atomic piece of software which is interconnected to other SWCs and BSW modules. Unlike the layered architecture style of an overall AUTOSAR frame-

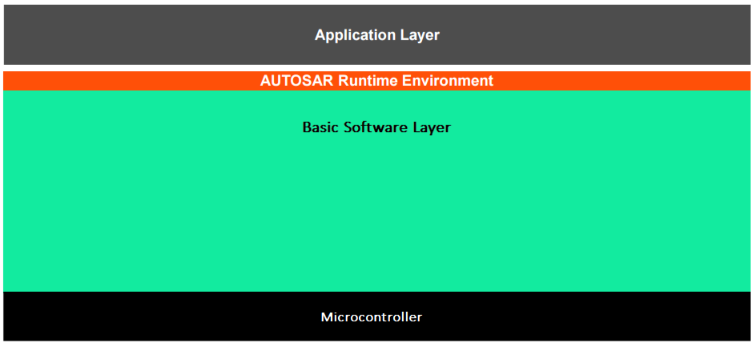


Figure 3.1: Layered software architecture for AUTOSAR

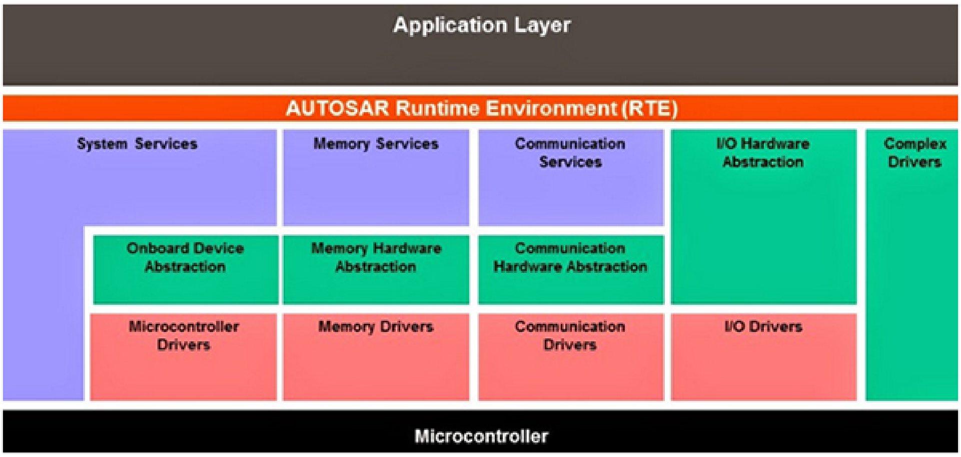


Figure 3.2: Classification of BSW layers. (Red - MCAL, Green - ECUAL, Purple - SL)

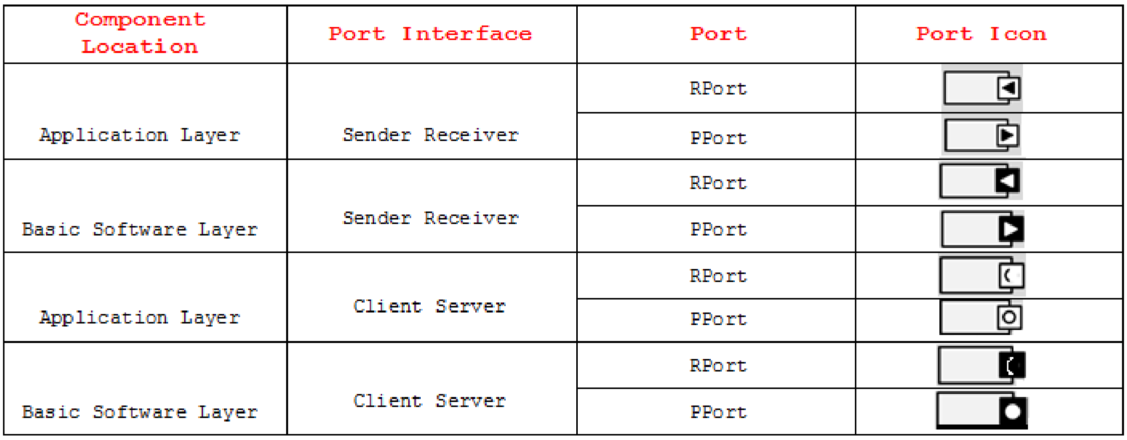
work, the application layer has a component architecture style. Adopting component style architecture enhances scalability and re-usability of SWCs.
Application SWCs makes use of ports and interfaces for communication. Two main kinds of ports are used i.e., Provide Port (PPort) and Request Port(RPort). Also, two main interface types are normally used, which are, client-server interface and sender receiver interfaces. The port notations are as shown in the Figure 3.3. Port notations depends on the placement of a SWC. A SWC can be placed in any layer (application layer or BSW layer) depending on the functionality it imparts. For example, if a SWC has client-server interface and is placed in BSW layer, then the PPort and RPort are represented accordingly by notations shown in Figure 3.3.

Application SWC can be of multiple types. Few important ones are as listed below.

1. Application SWC - An application SWC is a basic building block of an AUTOSAR application, as shown in Figure 3.4. It is used to carry out a particular application task within the system. It is a part of application layer and has no direct communication with the underlying BSW layer.

2. Parameter SWC - A parameter SWC is used to store the parameter values like calibration data, variables, fixed data etc required by the application and BSW. The main purpose of this SWC is to only provide data when requested and hence has only PPort as shown in Figure 3.5.

3. Sensor / Actuator SWC - Sensor / Actuator SWC (Figure 3.6) is used to interact with the ECU abstraction SWC directly in order to read the sensor values and access the actuators.



Component Location	Port Interface	Port	Port Icon
Application Layer	Sender Receiver	RPort	◁
		PPort	▷
Basic Software Layer	Sender Receiver	RPort	◀
		PPort	▶
Application Layer	Client Server	RPort	(
		PPort	◎
Basic Software Layer	Client Server	RPort	(
		PPort	●

Figure 3.3: Port notations

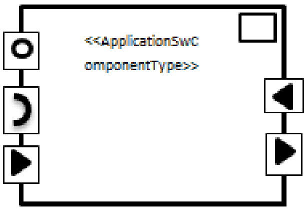


Figure 3.4: Application SWC type

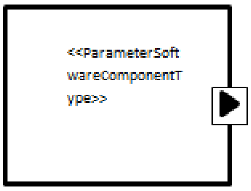


Figure 3.5: Parameter SWC type

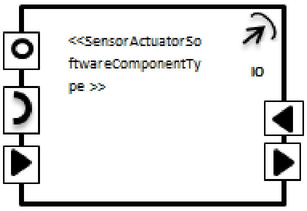


Figure 3.6: Sensor Actuator SWC type

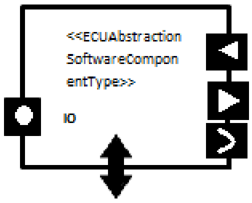


Figure 3.7: ECU abstraction SWC type

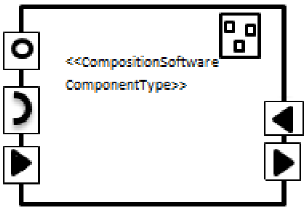


Figure 3.8: Composition SWC type

A sensor component is placed in application layer and has a client port to request sensor data from the underlying ECU abstraction SWC.

4. ECU abstraction SWC - An ECU abstraction SWC, unlike other SWCs, is present in the ECU abstraction layer (i.e within the IO hardware abstraction module in Figure 3.2). Port notations changes accordingly since this component is placed within BSW layer. This component can access ECUs I/O directly. It is the only SWC that has a direct access to BSW modules, as shown in Figure 3.7.

5. Composition SWC - A composition SWC (Figure 3.8) encapsulates one or more SWCs thereby providing abstraction from multiple applications on the same ECU. It encapsulates the SWCs of one application thereby providing a separation from SWCs of another application present on the same ECU.

Each application component defines an internal behavior for a particular component. An internal behavior specifies how a software component behaves with the rest of the architecture. It includes runnables that specifies the functionality of a SWC.

A runnable is a small software block that implements a certain function. This function is triggered by certain events called RTE Events. An RTE Event activates a runnable entity thereby addressing timing events, sending and receiving data (sender receiver) events, invoking operations, client server events, mode switching and other external events. For example, when the data is received over a port then *data received event* is generated by RTE that triggers the runnable entity which is responsible to receive the data. The data within a SWC is mapped to ports and the RTE layer establishes communication to other components and/or to BSW layer via ports and interfaces.

3.1.2 BSW Layer

Basic Software Layer provides core system functionality which consists of modules that imparts specific functionalities for communication, memory, IO etc. Sub-layers within the BSW layer (Figure 3.2) are explained as follows.

Services Layer (SL)

The Service Layer of basic software provides top level services to application software components. These services include operating system functionality, communication services, management services, memory services, ECU state management, mode management, diagnostic services to name a few.

ECU Abstraction Layer (ECUAL)

The ECU Abstraction Layer provides abstraction for drivers present in Micro-controller Abstraction Layer (MCAL). It also contains drivers for the external or on-board devices (off chip drivers). ECUAL masks the position of drivers (on chip or off chip) and provides an abstraction to the layers above. It offers an API for accessing the peripherals and devices regardless of their location and connection to the micro-controller and thus makes higher software layers independent of the hardware layout. ECUAL also includes the Complex Device Driver (CDD) layer. CDD is used for deploying functionality that is not available in other modules (e.g. proprietary software). CDD layer connects to the underlying hardware directly as shown in the Figure 3.9. Hence, applications that have hard deadlines can also be incorporated in this layer. The drivers implemented in this layer do not navigate through the AUTOSAR BSW layers and accesses the micro-controller directly.

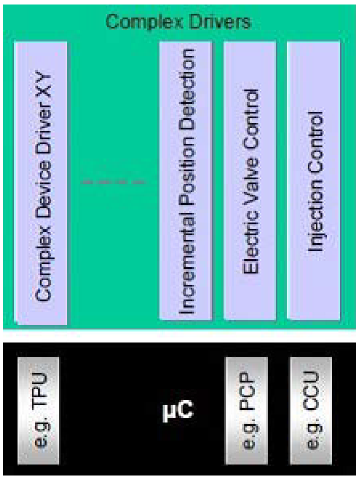


Figure 3.9: Complex Device Driver Layer

Micro-controller Abstraction Layer (MCAL)

Microcontroller Abstraction Layer is the lowest layer of abstraction. It contains internal drivers, which are driver modules that accesses the underlying micro-controller and internal peripherals directly, as shown in Figure 3.10. Internal devices are located inside the micro-controller like the internal Electrically Erasable Programmable Read-Only Memory (EEPROM), internal CAN driver etc. MCAL provides abstraction to the higher software layers and masks the underlying hardware details.

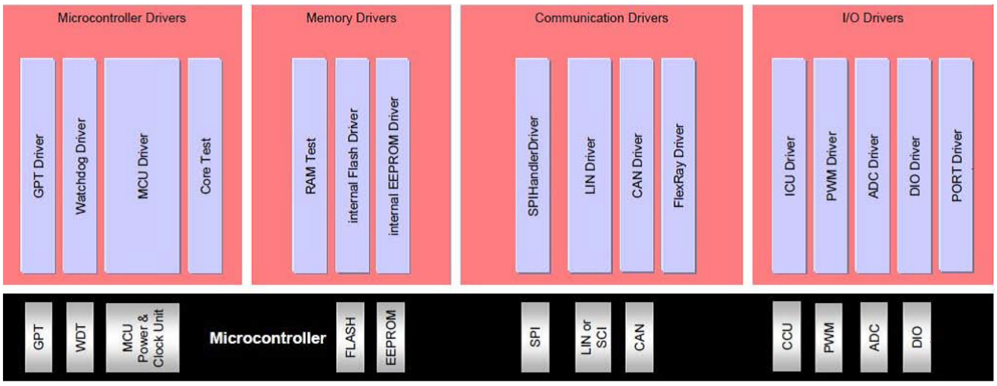


Figure 3.10: Microcontroller Abstraction Layer

3.1.3 RTE Layer

The Run-Time Environment (RTE) layer facilitates communication between the SWCs. It also creates a join between the application software and the underlying BSW layer via standardized interfaces. The RTE layer decouples the application layer from the hardware architecture. RTE in the realm of software development phase is known as Virtual Function Bus (VFB). VFB is the virtual implementation of RTE which provides similar features as a real RTE layer. VFB aids a developer in early testing of the application software as it provides standard services of the underlying system. RTE also maps the runnables to Operating System (OS) tasks. These runnables are triggered by events called the RTE Events. On occurrence of a certain RTE event (for example, timing event that triggers the runnable every 0.1s), the runnable gets executed and the necessary function is performed.

3.2 AUTOSAR Interfaces

The AUTOSAR interfaces provide standardized APIs between the application layer and BSW layer and between functional units within the BSW layer. These standardized interfaces facilitates software re-usability and interoperability. Three basic types of interfaces constitutes the AUTOSAR architecture as shown in the Figure 3.11 viz., AUTOSAR Interface, Standardized AUTOSAR Interface, Standardized Interface [40]. Black arrow indicates standardized interfaces which are relevant to VFB and RTE, usually required during the development of application layer and testing (virtual implementation). Yellow arrow indicates the interfaces relevant to RTE and are mainly used while configuring ECU and green arrows indicate the interfaces relevant to BSW modules that provides standardized APIs between modules.

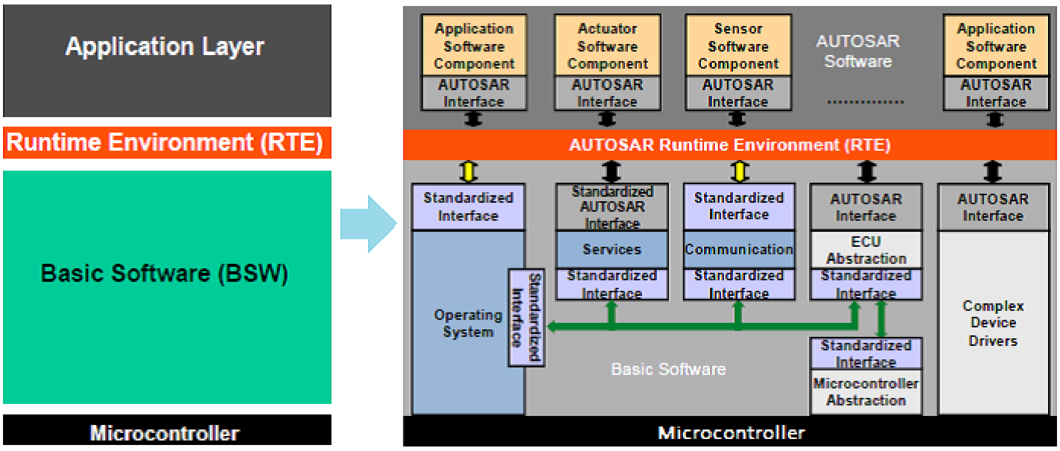


Figure 3.11: AUTOSAR Interfaces

1. AUTOSAR Interface
   An AUTOSAR interface, as shown in Figure 3.5, defines the information exchanged between software components and/or BSW layer. Client Server and Sender Receiver are the two interfaces that are commonly used and is independent of any programming language, ECU or network technology. AUTOSAR interfaces communicate via ports in SWCs. AUTOSAR makes it possible to implement this communication between software components and/or BSW modules either locally or via a network.

2. Standardized AUTOSAR Interface
   A Standardized AUTOSAR Interface is an AUTOSAR Interface whose syntax and semantics are standardized in AUTOSAR. The ”Standardized AUTOSAR Interfaces” are typically used to define AUTOSAR Services, which are standardized services provided by the AUTOSAR Basic Software to the application Software-Components.

3. Standardized Interface
   A Standardized Interface are APIs which are standardized within AUTOSAR BSW layer. These Standardized Interfaces are typically defined for a specific programming language (like ”C”). Because of this, standardized interfaces are typically used between software-modules which are always on the same ECU. When software modules communicate through a ”standardized interface”, it is not possible to route the communication between software-modules through a network.

3.3 AUTOSAR Methodology

AUTOSAR methodology, describes various steps followed in the process of implementing AUTOSAR architecture on an ECU. AUTOSAR adopts a uniform work-flow for the system development. All steps that are required to implement AUTOSAR architecture from system description to the generation of binaries are shown in Figure 3.12 [41]. System configuration input constitutes a system level design both for software, hardware and network architecture and other system level constraints (for example, number of cores, memory capacity etc.). This serves as the first artefact / input for developing an AUTOSAR system. Once the system is configured, the output is generated (System Configuration Description) and this artifact (.XML file) serves as an input to the next phase in the AUTOSAR methodology.

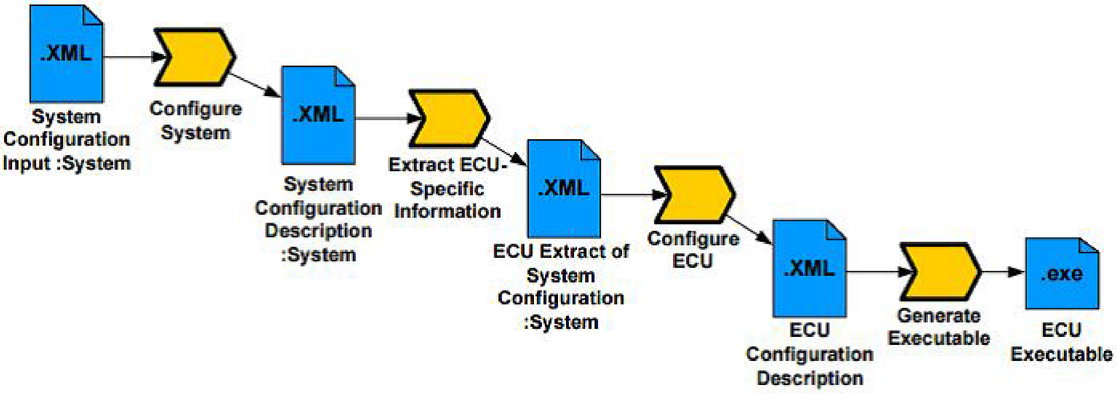

.XML
System Configuration Input :System
Configure System
.XML
System Configuration Description :System
Extract ECU-Specific Information
.XML
ECU Extract of System Configuration :System
Configure ECU
.XML
ECU Configuration Description
Generate Executable
.exe
ECU Executable

Figure 3.12: AUTOSAR Methodology

System Configuration Description contains the information necessary to configure an entire system. In the next phase, configuration details required for only one ECU is extracted which is termed as ECU Extract. An ECU extract includes a system configuration description for a specific ECU. This implies that a one to one mapping of the system configuration description for a particular ECU is made. It also includes application software configuration description (part of System Configuration Description). The output of this phase is an XML file (ECU Extract of System Configuration) and is the input for next phase of development.

Next, the required BSW modules and RTE layer are configured. In this phase, OS tasks are configured and the runnables are mapped to tasks. The configuration information of all modules along with RTE are stored in XML files (ECU configuration description). At this stage, the RTE generator also generates configuration files (.c and .h files).

Finally, the compiler compiles all the artifacts (.c and .h configuration files and source files) to generate an executable or a binary file. As a result, a .out / .elf file is generated as an executable that can be flashed on the micro-controller.

For each of these configuration steps, user interacts with the AUTOSAR tools for generating the corresponding artifacts explained in section 3.5.

3.4 Theoretical knowledge of the required BSW modules

Out of 63 BSW modules only those modules required for this application are configured, in order to reduce complexity. This section discusses only those required modules within the BSW layer which are used in implementation of the demonstrator (explained in Chapter 5). The main functionality of the demonstrator is to establish CAN communication stack. Figure 3.14 shows all the available modules within the AUTOSAR BSW layer, but only required modules for this application are explained as follows.

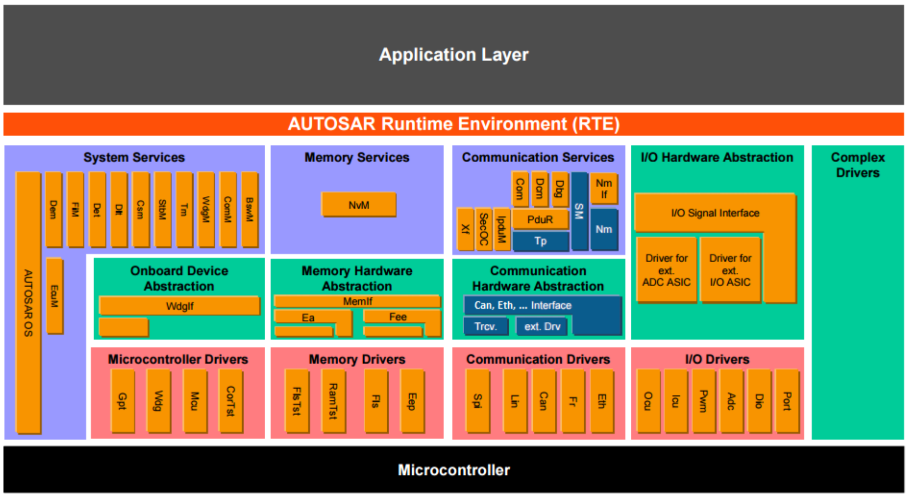


Figure 3.13: AUTOSAR BSW modules

1. COM
COM is the Communication Module (COM) that provides communication services. It predominantly manages the various signals received by it within the AUTOSAR architecture. It packs and unpacks the AUTOSAR signals from the RTE layer into Inter-network Protocol Data Units (I-PDUs) and the IPDUs received from the layers below into signals and provides those signals to the RTE, respectively. It also performs routing of signals from received IPDUs to the IPDUs that needs to be transmitted.

2. PduR
Protocol Data Unit Router (PduR) mainly manages the communication matrix/ routing table. It routes the IPdus to their respective communication stack i.e. if the received IPDU in the PDUR matches to a CAN-IPdu in the communication matrix, then that IPdu signal gets routed to the CAN communication stack. This is the most basic functionality of the Pdu Router module.

3. CANIF
CAN Interface (CANIF) module provides an abstraction to the CAN modules in services layer from the CAN driver module in the layers below. If its an external CAN device, then an external CAN driver is present within the ECUAL as shown in Figure 3.7, which then communicates to the hardware via SPI driver in the MCAL. If its an internal CAN hardware module then CANIF communicates with CAN driver within the MCAL layer.

4. CAN
CAN driver enables communication with the underlying CAN device. More information on how CAN communication works is presented in the appendix A1.

5. AUTOSAR OS
The operating system within an AUTOSAR architecture is OSEK compliant. It imparts basic OS functionality i.e manages the hardware resources like memory, real-time task scheduling, memory management, manages IO devices, interrupt management to name a few. AUTOSAR OS has some important entities which are counter, events, tasks, alarms, applications, interrupts, resources and schedule tables. All these entities perform similar

task of a traditional OS. Counter keeps the count in the OS module. Events are the entities that are referred by tasks. Tasks are the entities that are triggered by alarms. Alarm performs certain actions at a certain time (referenced by the counter). Interrupts are the special signals that halts the execution of the current task and temporarily executes a different task. Resources are the hardware resources and schedule tables which is used to schedule tasks based on their priorities. Here, application doesn't mean the application components. Within the AUTOSAR OS, application acts like a container for counters, tasks, events for that particular application. Hence, there can be more than one application running on ECU but within an AUTOSAR OS, the specific counters, tasks, events are bound by the application entity particular to that application.

6. ECUM
   ECU Manager (ECUM), handles the state of an ECU i.e. STARTUP, SHUTDOWN, RUN, SLEEP, WAKEUP. It acts like an ECU state machine manager which manages the state in which the ECU is currently in. By configuring ECUM to one or more states for different conditions an user can manage the control of an ECU. All AUTOSAR applications should include ECUM which provides ECU management services to the application components.

7. BSWM
   Basic Software Mode Manager (BSWM) is a module in the services layer of the AUTOSAR architecture. BSWM provides communication between different BSW modules and the application components. It mainly services the mode switch requests of the application via the RTE i.e. it takes care of BSW and application component's mode arbitration and mode control. Mode arbitration is based on some simple rules that have boolean logic i.e. a set of actions are determined which needs to be executed as a part of an action list and which should not be included in that list. The execution of the actions within the action list after arbitration is completely based on ECU configuration. The ECUM communicates with BSWM to notify ECU states and also notifies the wakeup source states.

8. MCU
   The MCU module is responsible for initialization of the microcontroller, clock other MCU modes specific to the hardware.

9. PORT
   PORT module in AUTOSAR provides drivers for initializing all the ports of the microcontroller (e.g. PORT A, PORT B etc.).

10. DIO
    The DIO module provides an abstraction to the microcontroller pins. It further allows grouping of these pins.

3.5 AUTOSAR tools

AUTOSAR tools are software tools that supports one or more tasks in the AUTOSAR methodology. Based on the functionality imparted by the tool-chain and considering the flow of the AUTOSAR methodology, it is found that four main categories of AUTOSAR tools exist.

1. Tools for modeling the system and application software and generate System Configuration Description artifact (see Figure 3.12).

2. Tools for coding and generating the Application SWCs (ASW) and generate ECU Extract of System Configuration.

3. Tools for configuring and generating the BSW layer and generate ECU Configuration Description.

4. Tools for generating the RTE layer and generate RTE files.

In Figure 3.12, modeling tool and ASW tool are used for generating system configuration description and ECU extract. BSW tool is used to configure ECU and generate ECU configuration description. RTE tool is used to generate RTE and configuration files. A tool vendor either provides a single tool-suite or multiple tools each performing one or more tasks. The flexibility in marketing these tools also provides a platform to add unique features that increases the profitability of these tools.

AUTOSAR tools from 23 vendors were considered initially out of which substantial information could be gathered for 13 tool-vendors. These vendors were further contacted for more information about their tool suite, though not all of them could be reached. As a result, some information were gathered from official tool website and other online resources. The prices for tools that were not known were extrapolated based on tool features, extent of coverage of the AUTOSAR methodology and other special features provided within the tool-suite, which are briefly explained:

1. ArcCore
   ArcCore's Arctic Studio provides tool that imparts modeling functionality for AUTOSAR SWCs [56]. It is based on ARText, a textual modeling language for defining ports, interfaces, software components, internal behavior and other elements. It is built on Eclipse platform and AUTOSAR tool platform called ARTOP [25] which imparts a common base functionality for AUTOSAR. It also facilitates the generation of ECU extract and configuration files in a specialized system language called ARXML. ARXML is an AUTOSAR XML file written in the standardized format and aids in data exchange. Although, ARText provides all the required features for modeling SWCs, Graphical User Interface (GUI) based model tools (model based design tools) works better in large applications and for this purpose ArcCore recommends using Mathworks's Embedded Coder and then importing the ECU extract into Arctic Studio.
   Artic Studio's ASW tool generates the 'C' code for application software from the ARText models. It also provides a development environment for editing C-code along with the required compilers for processing the code. This tool supports Software Component Description files in ARText format.
   Arctic Studio also provides BSW tool and RTE generation tool [57] [58]. These tools are used for ECU configuration and code generation. Arctic Core is ArcCore's proprietary software for BSW core modules which also includes configurations for selected micro-controller boards. Further, suitable compiler is also provided to generate binaries.
   Usability of a tool is evaluated based on intuitive features provided in the tool-suite. Arctic Studio mainly uses the interfaces provided by Eclipse platform throughout its development phase [67]. The tool includes a navigator panel to browse through all the elements within the ARXML packages.
   Arctic Studio provides a standard AUTOSAR data exchange formats like ARXML, and also supports CAN data exchange standards such as Field Bus Exchange (FIBEX), Data Base Container (DBC). However, nothing could be found on if the tool provides support for the integration of third party tools. Hence, using artifacts from other tool vendors might not be a straightforward approach.
   The generator tool in Arctic Studio can also validate the module configurations. It supports Xtend and Xpand framework for model validation. Xpand imparts logic to check if a given element is referenced by other elements thereby checking the dependencies between these elements. For example, to find all the signals that are not referenced by a certain PDU. However, the tool does not provide simulation and debugging features for an early testing and verification. As a result, its not entirely feasible to test the application using this tool.
   The cost for Artic Studio was quoted as €5.550,00 per unit price. The cost of Arctic Core standard package was quoted as €24.000,00 and the Arctic Core MCAL package as €7.200,00 per unit price. The total cost for the AUTOSAR tool-chain license from ArcCore would range about €50.000,00 (approx.) All the costs are quoted as of April 04 2017. Tool is distributed online and is available almost instantly after procuring the license.

2. COMASSO
   COMASSO is a community which contributes towards the development of open source (for a subscription fee) AUTOSAR BSW modules [48]. Robert Bosch GmbH initiated this community and also provides first set of BSW-Modules and Acceptance Test Software Components (AT-SWC) which is further improved by the members of the COMASSO community. In addition to the BSW core modules COMASSO also provides a tool for configuration and code generation. The tool does not provide any features apart from BSW core modules and configuration tool.
   COMASSO's BSW development tool is also built on Eclipse platform [60]. Apart from Eclipse GUI nothing could be found about the extent of intuitiveness adopted in this tool. Tool provides AUTOSAR standard data exchange format and also supports CAN data exchange standards such as FIBEX, DBC. The tool makes use of Xpand framework for model validation [59].
   The cost for becoming a member for this community was quoted as €1.000,00 for 1st year membership and from 2nd year on-wards €2.500,00 per year.

3. Continental
   Continental's AUTOSAR tool-suite covers all stages of development. It also provides a BSW and RTE configuration tool called CESSAR-CT. The tool allows modeling a self contained AUTOSAR architecture. It was also noted by some users that the usability of tool had to be improved when the tool is applied on a large system [61]. Continental's website does not provide much information about their AUTOSAR tool-suite. It might also be the case that this vendor has other channels to promote their solution and doesn't provide direct sales of AUTOSAR tools.
   Like ArcCore, CESSAR-CT is based on Eclipse and ARTOP framework. CESSAR-CT provides a plugin mechanism such that the tool user can extend the tool functionality. Code generators of different technologies can be integrated and a form editor enables the extensibility and customization of UI [66]. The tools are interoperable and support standard data exchange formats.

4. Dassault systems
   The AUTOSAR tool provided by Dassault Systems is called the AUTOSAR Builder. It is a an open authoring and simulation tool that enables rapid modeling, definition, simulation and deployment of embedded systems to automotive ECUs [62]. Like other tools, this is also based on Eclipse platform for the design and development of AUTOSAR compliant systems and software and extensible and customizable based on ARTOP tool platform. The Builder tool is full-fledged and delivers a set of dedicated development environments that fully supports all the stages of AUTOSAR development process [63]. AUTOSAR Builder is a part of the CATIA Systems Engineering solution from Dassault Systems. The tool provides high level GUI and intuitive features like simulation. Further, the tool provides support for integration of legacy software and aids in migration to AUTOSAR. Integration of third party tools enables interoperability between modeling languages and code generation tools. Also facilitates early testing and verification. The approximate cost of CATIA tool with AUTOSAR builder is around €60,000 to €65,000.

5. dSPACE
   dSPACE's development tool suite consists of SystemDesk and TargetLink [64] [65]. SystemDesk tool aids in modeling, network architecture design and RTE generation, as per the AUTOSAR methodology. The RTE generation module is available as an add-on for SystemDesk tool, which generates full-fledged RTE for communication of SWCs with the ECU. TargetLink is application code generator tool which generates production code from the graphical representation of the architecture modeled in SystemDesk or a third party graphical environment tool like MATLAB/Simulink. The tool-suite also includes a simulator, which simulates systems that span across one/more ECUs [62]. However, the tool-suite does

not provide BSW modules. The tool also provides intuitive features and high level GUI. TargetLink imparts interoperability and early testing.

6. Elektrobit
Elektrobit (EB) tool-suite mainly focuses on providing RTE, BSW and other low level services. It's EB Tresos tool-suite includes EB Tresos designer, EB Tresos Studio, EB Tresos AutoCore [12]. Apart from these they also provide tool that incorporates functional safety standard ISO26262 called EB Tresos Safety. EB does not provide tools or services to develop application and modeling part of the AUTOSAR implementation. The designer tool is mainly used to design the network architecture of the system which can also extract a communication matrix (part of system configuration input in the AUTOSAR methodology). EB Tresos Studio is also based on eclipse platform and provides, configuration, validation and generation of the basic software. The Tresos AutoCore is the system software that includes BSW modules which are AUTOSAR standard compliant. AutoCore is EBs BSW core and consists of more than 40+ BSW modules which are configured using EB Tresos Studio tool. With recent developments in autonomous cars and automated driving, EB also provides another tool called EB assist which is used to develop driver assistance systems with increased safety features [68].

7. ETAS
A set of AUTOSAR tool solutions provided by ETAS includes ISOLAR-A, ISOLAR-EVE, RTA, ASCET and COMASSO [13]. ISOLAR-A tool is used to generate AUTOSAR system and application design, configure ECU and generate RTE. This tool also facilitates the integration of third-party tools. ISOLAR-EVE on the other hand provides a virtual ECU environment facilitating early testing, verification and validation. This is nothing but Virtual Function Bus (VFB) where early testing of application SWCs can be done without deploying it on an actual hardware. ASCET-DEVELOPER tool (also known as ASCET 7) is a tool for developing application software for embedded systems using graphical models and textual programming notations. It also provides code generation functionality. The RTA family tool-suite consists of RTA-OS, RTA-RTE and RTA-BSW tools. RTA-OS is the real-time operating system, RTA-RTE is the AUTOSAR run-time environment generator and RTA-BSW provides all the AUTOSAR BSW modules in compliance with the functional safety standard ISO26262. RTA-BSW tool also provides MCAL modules for specific microcontroller hardware. Since ETAS and COMASSO have same parent organization i.e. Bosch GmbH, the BSW modules form COMASSO are supported within the ETAS development environment. ETAS provides rich documentation and knowledge transfer about their AUTOSAR tool-suite and also enables integration of other third-party tools via standardized interfaces.

8. KPIT
The K-SAR tool-suite from KPIT provides AUTOSAR BSW modules, BSW configuration tool, MCAL for selected microcontrollers, RTE code generation and K-SAR editor for AUTOSAR v3.X and v4.X compatible ECUs. KPIT does not provide tools that entirely cover the development of application although it supports the integration of third party tools at each stage of development [69]. Its main area of expertise is in developing MCAL drivers as a part of AUTOSAR stack, developing BSW modules and to provide services around these modules. Other key features of this tool-suite are it provides multi-core support, end to end communication protection, support for integration of third party tools and other safety critical features.

9. Mathworks
MATLAB Embedded Coder tool provides AUTOSAR support, using which we can generate AUTOSAR compliant C / C++ code and export in ARXML format [70]. It also integrates AUTOSAR support within Matlab/Simulink using which application modeling can be done. It also serves as a 3rd party modeling tool for other tool vendors. Hence, Mathwork only provides AUTOSAR application layer features. The price for Embedded Coder tool (as found on internet) was approximately €6500 as of April 27 2017.

10. Mecel
    Mecel Picea, which is now a part of Mentor Graphics, provides AUTOSAR tool solutions called Mecel Picea Suite and Mecel Picea Workbench [72]. Mecel Picea suite includes AUTOSAR BSW modules and the workbench includes BSW configuration and RTE generation tool. These tools are based on the Eclipse and ARTOP platform. Picea and the tools provided by Mentor Graphics are developed in such a way that they can be integrated with each other. From the research it seems like Mecel Picea tool-suite provides less coverage for modeling and application development although support for some features like modeling network architecture exists. Picea provides minimal support for integration of third party tool. It meets the requirements for functional safety levels Automotive Safety Integrated Level (ASIL) from level A to level D, as described in ISO26262.

11. MentorGraphics
    The Volcano family tool-suite provided by Mentor Graphics includes Volcano VSTAR, Volcano VSB and Volcano VSI [73]. Volcano VSTAR provides BSW modules, MCAL modules for specific microcontroller hardware, ECU configuration and RTE generation functionalities. This tool also meets requirements of the functional safety standard ISO26262. Volcano Vehicle System Builder (VSB) tool imparts designing of SWCs and integration of SWCs with basic software. It also provides services like network design, diagnostics and database management. Volcano Vehicle System Integrator (VSI) enables virtual software integration and early testing. Mentor Graphics together with Mecel (acquired by Mentor Graphics) provide full-fledged tool solutions for implementing AUTOSAR methodology.

12. Opensynergy
    The COQOS tool from Opensynergy is a hypervisor and runs on Linux platform or other POSIX operating system. Its central technology is virtualization [71]. COQOS hypervisor enables the creation of Virtual Machines (VMs) upon which both general purpose operating system like Linux and real-time operating system like AUTOSAR OS can function simultaneously and also communicate with each other. COQOS SDK consists of COQOSAR OS along with BSW scheduler embedded within a virtual machine. COQOSAR also incorporates Opernsynergy's Automotive Communication Framework (ACF) as a part of integration with the CAN communication stack. COQOS SDK does not provide any other BSW modules except the OS and scheduler which integrates software components and other third party BSW modules into on-board network. The security provided by the COQOS tool ensures that the guest operating system runs independently and thus the partition acts as a firewall and provides protection against external attacks. The COQOS SDK further consists of tools for configuring the BSW modules and generating RTE layer [71]. Opensynergy does not support the application and modeling phases of AUTOSAR development.

13. Vector Informatik GmbH
    Vector Informatik provides modeling tool called PREEvision which is used to design SWCs and network architecture. DaVinci Developer tool along with DaVinci Configurator Pro tools are used to develop BSW modules, configure ECU and generate RTE. VIRTUAL target tool enables virtual implementation of SWCs facilitating early verification and testing. All these tools meet requirements of functional safety standard ISO26262 [11]. For simple ECUs which are less powerful, Vector provides MICROSAR as an Operating System which acts like a lightweight AUTOSAR OS. It further provides integration of third-party BSW and MCAL modules. The prices quoted by Vector were, PREEvision tool ranges from around €55.000,00 upto €200.000,00 (floating license for team collaboration mode). DaVinci Configurator Pro was quoted as €8.700,00. DaVinci Developer €9.050,00. Both DaVinci Developer and Configurator tools totally costs about €21.153,44. All the prices were quoted as of date April 12 2017.

Table 3.1 summarizes all the AUTOSAR tool features discussed above.

Table 3.1: AUTOSAR tool features

Tool vendors	**Tools**	**BSW / MCAL code implementation**	**BSW configurator tool**	**RTE generator tool**	**SWC implementation tool**	**System and Software modeling tool**	**Other features**	**License**
ArcCore	Arctic Core, Arctic Studio	✓	✓	✓	✓	✓	Built on Eclipse platform and AUTOSAR tool platform called ARTOP	GPL, Commercial, Evaluation license
Comasso	COMASSO_4.0.2.x, BSWDT	✓	✓	✗	✗	✗	Built on Eclipse platform and ARTOP platform	Community
Continental	Continental AUTOSAR tool-suite, CESSAR-CT	✓	✓	✓	✓	✓	Built on Eclipse platform and ARTOP platform	Commercial
Dassault Systems	AUTOSAR Builder tool suite (AAT, GCE, RTEG, ASIM, ART)	✗	✓	✓	✓	✓	Rapid modeling, Definition and simulation tool-set, Built on Eclipse platform and ARTOP platform, Enables import of model-based design legacy descriptions and generation of AUTOSAR compliant code, Enables integration with 3rd party tools to support interoperability, Early verification of ECUs	Commercial

dSPACE	SystemDesk, TargetLink	✗	✗	✓	✓	✓	Virtual ECU simulation, Rapid prototyping, Certification for ISO 26262, ISO 25119, IEC 61508 and derivative standards, Validation and verification	Commercial
Elektrobit	EB Tresos AutoCore, EB tresos Studio	✓	✓	✓	✗	✗	Certification for ISO 26262 upto ASIL D, Enables integration with 3rd party tools to support interoperability, Built on Eclipse platform	Commercial
ETAS	ISOLAR-A, ASCET, RTA-BSW, RTA-RTE, COMASSO	✓	✓	✓	✓	✓	Certification for IEC 61508, ISO 26262 upto ASIL D, Built on Eclipse platform and ARTOP platform, Enables integration of third-party tools via open and standardized interfaces	Commercial
KPIT	K-SAR AUTOSAR Suite	✓	✓	✓	✗	✗	Built on Eclipse platform and ARTOP platform, support for integration of third party tools	Commercial
Mathworks	Embedded Coder	✗	✗	✗	✓	✓	Serves as a third party tool for modeling and generating the code for application layer	Commercial
Mecel (acquired by Mentor Graphics)	Picea	✓	✓	✓	✓	✓	Built on Eclipse platform and ARTOP platform	Commercial
Mentor Graphics	Volcano System Architect, Volcano VSTAR	✓	✓	✓	✓	✓	Certification for ISO 26262 upto ASIL D, Built on Eclipse platform and ARTOP platform	Commercial

Opensynergy	COQOS suite (SDK, Hyperviser)	✓	✓	✓	✗	✗	Provides hardware virtualization, COQOS hypervisor creates Virtual Machines, Provides support for ISO 26262 standard	Commercial
Vector Informatik Gmbh	PREEvision, DaVinci Developer, DaVinci Configurator Pro, v VIRTUALtarget	✓	✓	✓	✓	✓	Enables integration with 3rd party tools to support interoperability, ISO 26262 certification, Virtual development and testing of AUTOSAR software, Early verification of ECUs	Commercial

Table 3.2 presents a brief description of each of these tools based on the 6 key criteria that were identified for the application of AHP algorithm. More on this can be found in Chapter 4. Table 3.3 presents the extent of support provided by each of these tools. Eventhough a tool provides a certain feature, it might not be full-fledged.

Table 3.1 gives an overview of the tools considered for comparison and what phases of AUTOSAR methodology are covered within each tool. Table 3.2 presents a brief description of each of these tools based on the 6 key criteria that were identified for the application of AHP algorithm. More on this can be found in Chapter 4. Table 3.3 presents the extent of support provided by each of these tools. Eventhough a tool provides a certain feature, it might not be full-fledged.

Table 3.3: AUTOSAR tool features coverage

Tool vendors	**Support for MCAL layer modules**	**Support for BSW code generation / configuration**	**Support for RTE generation**	**Support for ASW code generation**	**Support for AUTOSAR modeling**
ArcCore	🟩	🟩	🟩	🟩	🟩
Comasso	🟩	🟩	🟥	🟥	🟥
Continental	🟢	🟩	🟩	🟢	🟢
Dassault Systems	🟥	🟩	🟩	🟩	🟩
dSPACE	🟥	🟥	🟩	🟩	🟩

Elektrobit	🟩	🟩	🟩	🟥	🟥
ETAS	🟩	🟩	🟩	🟩	🟩
KPIT	🟩	🟩	🟩	🟥	🟥
Mathworks	🟥	🟥	🟥	▧	🟩
Mecel	▧	🟩	🟩	▧	▧
Mentor Graphics	🟩	🟩	🟩	🟩	🟩
Opensynergy	▧	🟩	🟩	🟥	🟥
Vector Informatik Gmbh	🟩	🟩	🟩	🟩	🟩

🟩	Full coverage	▧	Partial coverage	🟥	Not supported

Chapter 4

Tools Selection Methodology

4.1 Overview

This chapter presents a methodology to select AUTOSAR tool-chain by applying AHP algorithm which was discussed in Chapter 2. Figure 4.1 gives a better understanding of the AUTOSAR methodology using V-model and also shows various tools required at each development stage of the AUTOSAR architecture.

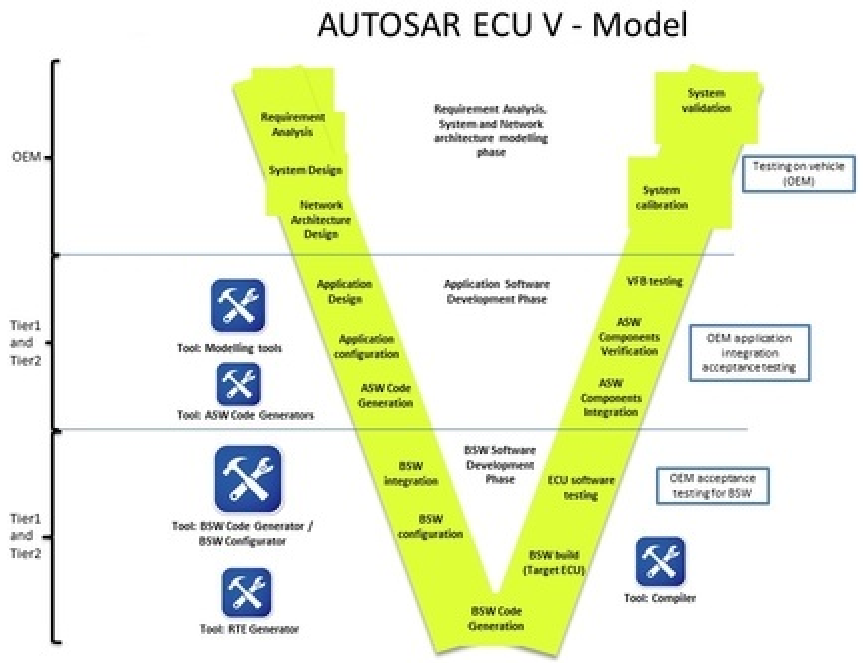

AUTOSAR ECU V - Model
OEM
Tier1 and Tier2
Tier1 and Tier2
Requirement Analysis
System Design
Network Architecture Design
Requirement Analysis, System and Network architecture modelling phase
System validation
System calibration
Testing on vehicle (OEM)
Application Design
Application configuration
ASW Code Generation
Application Software Development Phase
VFB testing
ASW Components Verification
ASW Components Integration
OEM application integration acceptance testing
Tool: Modelling tools
Tool: ASW Code Generators
BSW integration
BSW configuration
BSW Software Development Phase
ECU software testing
BSW build (Target ECU)
BSW Code Generation
OEM acceptance testing for BSW
Tool: BSW Code Generator / BSW Configurator
Tool: RTE Generator
Tool: Compiler

Figure 4.1: Architecture Design in Software Development Cycle

Accordingly, AUTOSAR tools can be classified into four main categories which are:

- Modeling tool
- Application code generation tool
- BSW generation and configuration tool (also includes the MCAL layer)

- RTE generation tool

Since the focus was to select a tool-chain for developing AUTOSAR, a tool selection methodology was put forth as shown in Figure 4.2. Each step in the methodology is applied individually for different tool categories mentioned above. At first the stakeholders are identified and their requirements are analyzed. Next, with the help of architecture design models, how each stakeholder interacts with the system is understood. A list of criteria is generated and the AHP algorithm is applied to the hierarchical list of criteria. A final AUTOSAR tool-chain selection is then made.

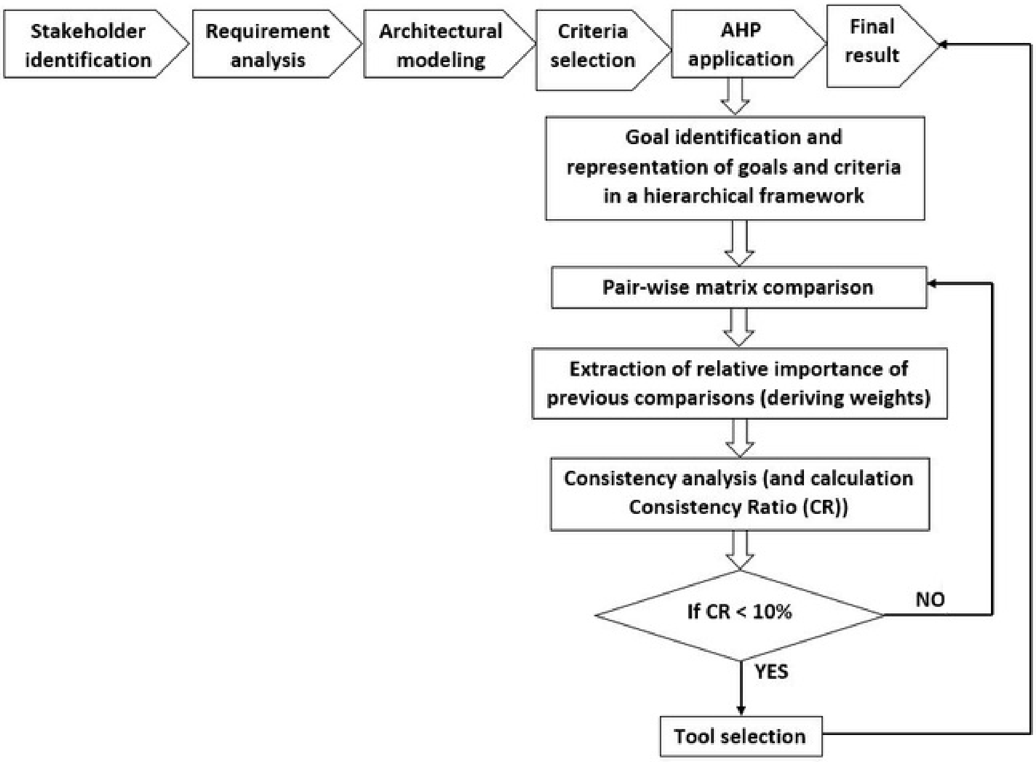

Stakeholder identification
Requirement analysis
Architectural modeling
Criteria selection
AHP application
Final result
Goal identification and representation of goals and criteria in a hierarchical framework
Pair-wise matrix comparison
Extraction of relative importance of previous comparisons (deriving weights)
Consistency analysis (and calculation Consistency Ratio (CR))
If CR < 10%
NO
YES
Tool selection

Figure 4.2: Proposed tool selection methodology work-flow

4.2 Stakeholder identification (Users of AUTOSAR tools)

The following stakeholders are considered in perspective of users who use the AUTOSAR tool-chain, i.e. Figure 4.1 presents different stages where the tool is applied or used to develop an AUTOSAR ECU. Here, we are analyzing the requirements / concerns of these users. This step was necessary because, we had to identify what were the minimum requirements that a stakeholder or an user has while using a tool to develop an automotive ECU based on the AUTOSAR architecture. As mentioned in Chapter 1, AUTOSAR tool-vendors add multiple features in their tools to make it profitable and hence each tool has its own share of pros and cons. The idea here was to eliminate all the unnecessary requirements and gain an insight of the minimum requirements that qualifies an AUTOSAR tool-chain which is explained in Section 4.3.

Stakeholders in this context were identified upon discussions with Engineers from Brace Automotive and Orlaco. Through these discussions, the most important concerns and requirements of both Engineers and Management of the companies, with regards to deploying AUTOSAR architecture in software development, could be identified.

1. Requirement analyst for AUTOSAR tool-chain
   A requirement analyst gathers all the requirements from other stakeholders involved in de-

veloping an ECU and performs an analysis taking into account possible conflicting requirements. These requirements must be actionable, measurable, testable, traceable and defined in a sufficient detail for system design, which is then documented. Requirements can be documented in various forms, usually in the form of a requirement diagram. Use cases, user stories, process specifications and variety of other models can be used to further analyze the use of these requirements. A Requirement Analyst, typically makes use of AUTOSAR modeling tool in the development of ECUs.

2. System designer (ECU)
   The system designer has a big picture of the entire system, who mainly defines the architecture, modules, interfaces, data and network signals for a system (ECU) to satisfy specified requirements. System designer works closely with the requirement analyst, software designers, application developers, system developers and other stakeholders.

3. Software designer
   Software designer designs software application and writes software code. In this case, the designer works closely with the system designer and requirement analyst to understand the high level functionality of the application on a given system. A software designer uses both model tool and application code generation tool for developing AUTOSAR ECUs.

4. Application developer
   Very often, there is no difference between a software designer and a developer, but in large scale applications, application/software developers are involved whose work mainly revolves around writing software code. An application developer mainly uses application code generation tool (though its not just that).

5. System developer (the one who writes the code for ECU)
   System developer works on system software namely operating system, drivers and other modules. They often work on low-level languages like C/C++ whose focus lies on creating a stable and reliable system software that can be used to build an application on. The system developer mainly uses BSW generation tool and configurator tool in the development of AUTOSAR ECUs.

6. System engineer
   System engineer is responsible for ensuring the system is configured to meet the stakeholders requirements. In this case, configuration of ECUs along with RTE generation is taken care of by the system engineer. The system engineer also works closely with the system developer to be able to configure the system according to the requirements and hence uses BSW generation and configuration tool along with RTE generator tool.
   After development of ECUs, in the testing phase, a tester typically performs unit testing of each module and integration testing of all the ECUs. Finally the functioning of ECUs are verified and validated to see if it conforms to the requirements set by OEM as shown in Figure 4.1. In this work testing tools are not considered, instead the testing capabilities of the tools used above are taken into account. This is to also quantify the tools in terms of early testing capabilities which is highly crucial in automotive industry. Thus **RQ2.1** is answered.

4.3 Requirements analysis

In the previous section, stakeholders were identified. Upon discussions with these stakeholders from Brace and Orlaco, their requirements, concerns and constraints were considered. These requirements were further fine tuned by analyzing the AUTOSAR standard specifications [17]. All the architectural diagrams are based on System Modeling Language (SysML) and IBM Rhapsody

SysML tool was used to create these models. Figure 4.3 shows a consolidated stakeholder requirement diagram for all the four tool categories which are further expanded in Figures 4.4 to 4.7.

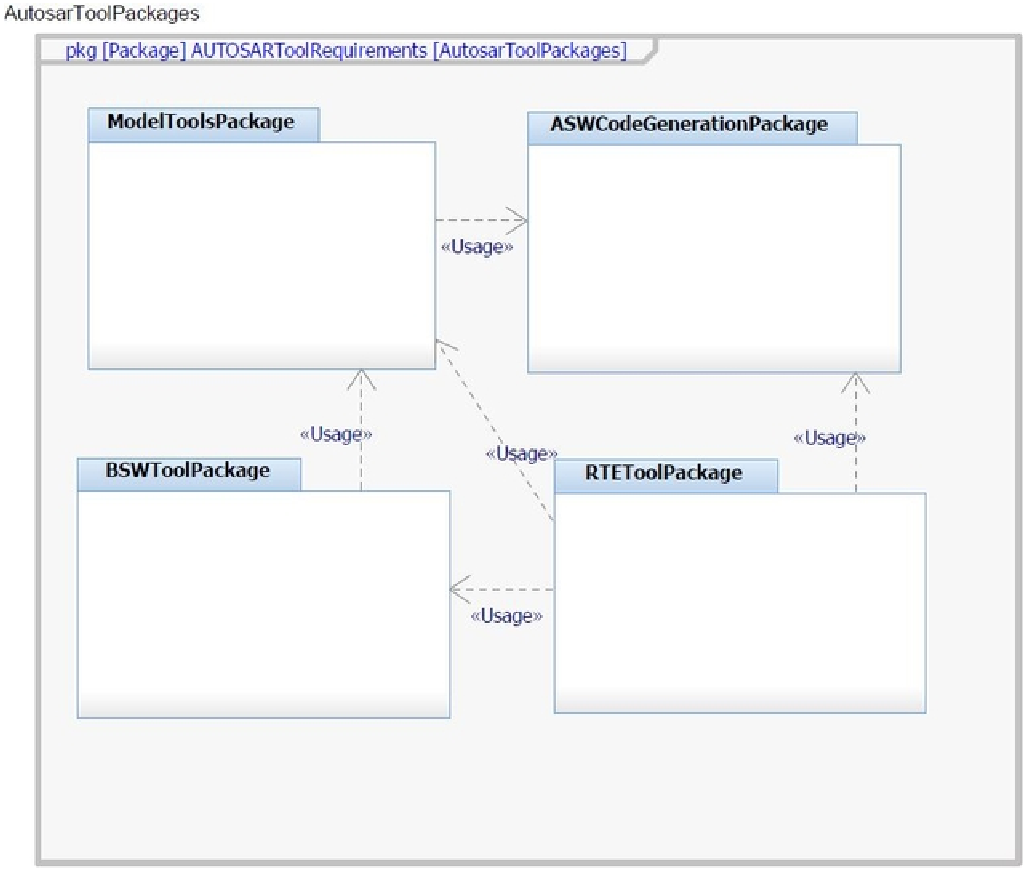
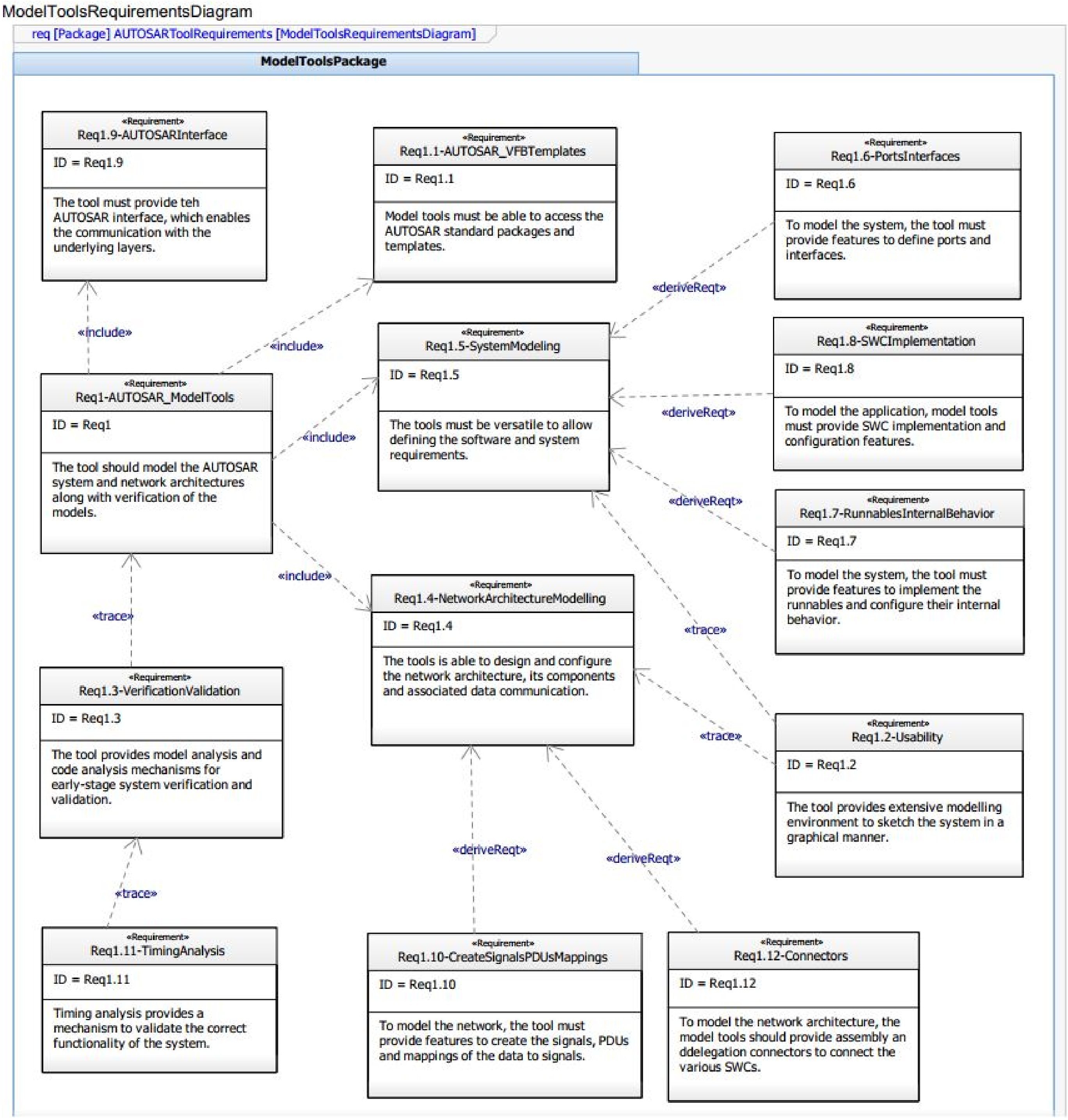


Figure 4.3: Consolidated stakeholder requirements diagram

<Usage> relation between the model indicates that the output from a specific tool is used as an input to the next tool (i.e output from ASW code generator tool i.e. C code for the application, is used by the model tool to generate ECU extract). Likewise, RTE makes use of the output from the model tool, ASW code generator tool and BSW tool in order to generate a glue between the application layer and the underlying BSW layer.
In Figure 4.3, RTE tool package has a <Usage> relation directed from all the other tool packages. This is because RTE generator tool compiles all the source files, configuration files, application files and other intermediary files in order to generate files configured by RTE.
<include> relation simplifies a large use case by splitting it into several use cases. It is a directed relationship between two use cases which is used to show the behavior of the included use case (the addition) is inserted into the behavior of the including (the base) use-case (main requirement has a number of sub-requirements and the satisfaction of all the sub-requirements automatically satisfies the main requirement).
<deriveReqt> relationship is used to represent a relationship between requirements at the same level of the hierarchy but at different levels of abstraction. For example, in Figure 4.4, the system modeling requirements of the model tools are further analyzed in order to derive more detailed requirements such as implementing SWC, runnables, ports and interfaces etc., that reflect additional implementation considerations or constraints.



«Requirement»
Req1.9-AUTOSARInterface
ID = Req1.9
The tool must provide teh AUTOSAR interface, which enables the communication with the underlying layers.

«Requirement»
Req1.1-AUTOSAR_VFBTemplates
ID = Req1.1
Model tools must be able to access the AUTOSAR standard packages and templates.

«Requirement»
Req1.6-PortsInterfaces
ID = Req1.6
To model the system, the tool must provide features to define ports and interfaces.

«Requirement»
Req1-AUTOSAR_ModelTools
ID = Req1
The tool should model the AUTOSAR system and network architectures along with verification of the models.

«Requirement»
Req1.5-SystemModeling
ID = Req1.5
The tools must be versatile to allow defining the software and system requirements.

«Requirement»
Req1.8-SWCImplementation
ID = Req1.8
To model the application, model tools must provide SWC implementation and configuration features.

«Requirement»
Req1.7-RunnablesInternalBehavior
ID = Req1.7
To model the system, the tool must provide features to implement the runnables and configure their internal behavior.

«Requirement»
Req1.4-NetworkArchitectureModelling
ID = Req1.4
The tools is able to design and configure the network architecture, its components and associated data communication.

«Requirement»
Req1.3-VerificationValidation
ID = Req1.3
The tool provides model analysis and code analysis mechanisms for early-stage system verification and validation.

«Requirement»
Req1.2-Usability
ID = Req1.2
The tool provides extensive modelling environment to sketch the system in a graphical manner.

«Requirement»
Req1.11-TimingAnalysis
ID = Req1.11
Timing analysis provides a mechanism to validate the correct functionality of the system.

«Requirement»
Req1.10-CreateSignalsPDUsMappings
ID = Req1.10
To model the network, the tool must provide features to create the signals, PDUs and mappings of the data to signals.

«Requirement»
Req1.12-Connectors
ID = Req1.12
To model the network architecture, the model tools should provide assembly an ddelegation connectors to connect the various SWCs.

Figure 4.4: Stakeholder requirements diagram for Model Tools

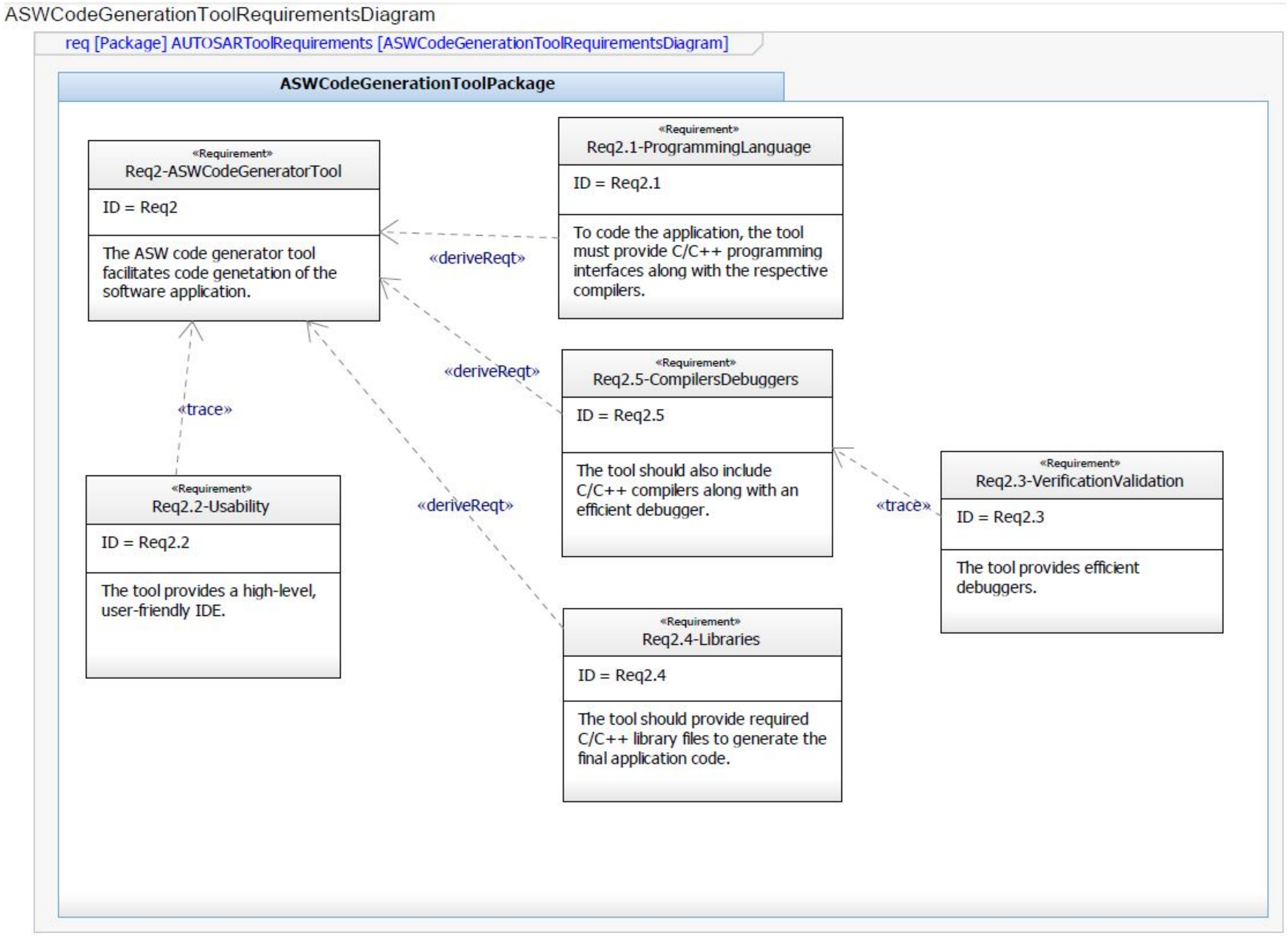

ASWCodeGenerationToolRequirementsDiagram
req [Package] AUTOSARToolRequirements [ASWCodeGenerationToolRequirementsDiagram]
ASWCodeGenerationToolPackage
«Requirement»
Req2-ASWCodeGeneratorTool
ID = Req2
The ASW code generator tool facilitates code genetation of the software application.
«Requirement»
Req2.1-ProgrammingLanguage
ID = Req2.1
To code the application, the tool must provide C/C++ programming interfaces along with the respective compilers.
«deriveReqt»
«deriveReqt»
«trace»
«Requirement»
Req2.5-CompilersDebuggers
ID = Req2.5
The tool should also include C/C++ compilers along with an efficient debugger.
«Requirement»
Req2.2-Usability
ID = Req2.2
The tool provides a high-level, user-friendly IDE.
«deriveReqt»
«trace»
«Requirement»
Req2.3-VerificationValidation
ID = Req2.3
The tool provides efficient debuggers.
«Requirement»
Req2.4-Libraries
ID = Req2.4
The tool should provide required C/C++ library files to generate the final application code.

Figure 4.5: Stakeholder requirements diagram for ASW Tools

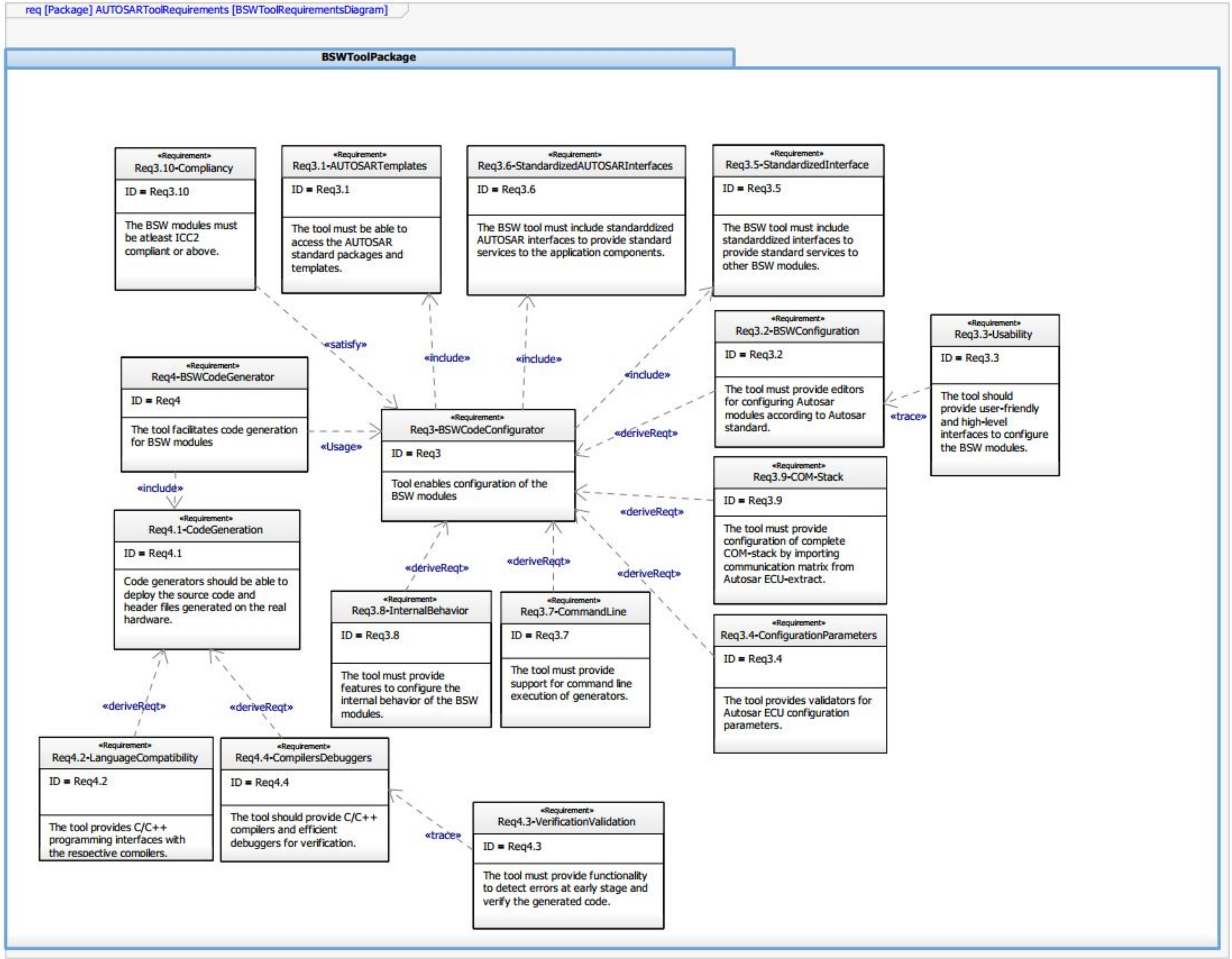
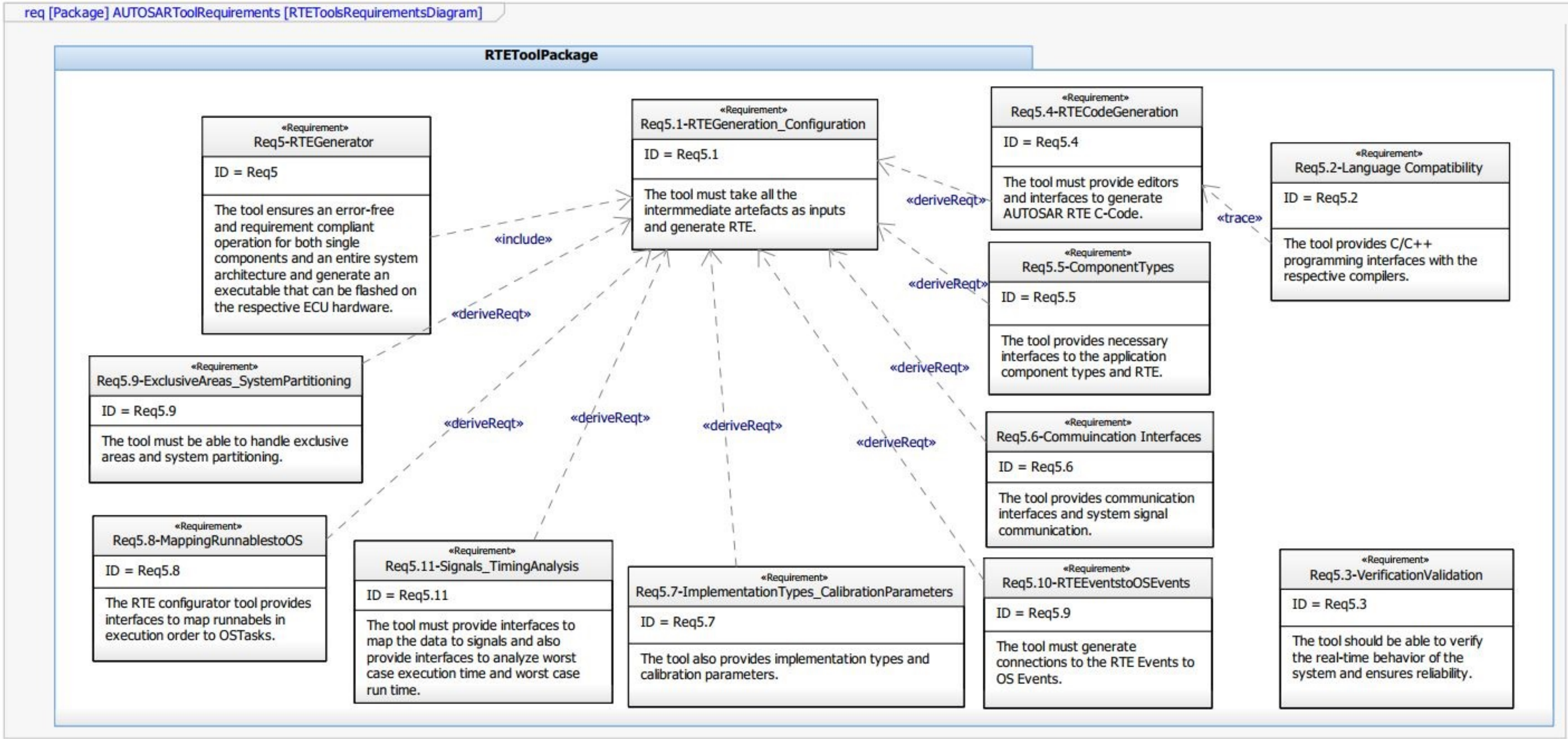

req [Package] AUTOSARToolRequirements [BSWToolRequirementsDiagram]
BSWToolPackage
«Requirement» Req3.10-Compliancy
ID = Req3.10
The BSW modules must be atleast ICC2 compliant or above.
«Requirement» Req3.1-AUTOSARTemplates
ID = Req3.1
The tool must be able to access the AUTOSAR standard packages and templates.
«Requirement» Req3.6-StandardizedAUTOSARInterfaces
ID = Req3.6
The BSW tool must include standarddized AUTOSAR interfaces to provide standard services to the application components.
«Requirement» Req3.5-StandardizedInterface
ID = Req3.5
The BSW tool must include standarddized interfaces to provide standard services to other BSW modules.
«Requirement» Req3.2-BSWConfiguration
ID = Req3.2
The tool must provide editors for configuring Autosar modules according to Autosar standard.
«Requirement» Req3.3-Usability
ID = Req3.3
The tool should provide user-friendly and high-level interfaces to configure the BSW modules.
«Requirement» Req4-BSWCodeGenerator
ID = Req4
The tool facilitates code generation for BSW modules
«Requirement» Req3-BSWCodeConfigurator
ID = Req3
Tool enables configuration of the BSW modules
«Requirement» Req3.9-COM-Stack
ID = Req3.9
The tool must provide configuration of complete COM-stack by importing communication matrix from Autosar ECU-extract.
«Requirement» Req4.1-CodeGeneration
ID = Req4.1
Code generators should be able to deploy the source code and header files generated on the real hardware.
«Requirement» Req3.8-InternalBehavior
ID = Req3.8
The tool must provide features to configure the internal behavior of the BSW modules.
«Requirement» Req3.7-CommandLine
ID = Req3.7
The tool must provide support for command line execution of generators.
«Requirement» Req3.4-ConfigurationParameters
ID = Req3.4
The tool provides validators for Autosar ECU configuration parameters.
«Requirement» Req4.2-LanguageCompatibility
ID = Req4.2
The tool provides C/C++ programming interfaces with the respective compilers.
«Requirement» Req4.4-CompilersDebuggers
ID = Req4.4
The tool should provide C/C++ compilers and efficient debuggers for verification.
«Requirement» Req4.3-VerificationValidation
ID = Req4.3
The tool must provide functionality to detect errors at early stage and verify the generated code.
«satisfy»
«include»
«include»
«include»
«trace»
«deriveReqt»
«Usage»
«include»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«trace»

Figure 4.6: Stakeholder requirements diagram for BSW Tools



«Requirement»
Req5-RTEGenerator
ID = Req5
The tool ensures an error-free and requirement compliant operation for both single components and an entire system architecture and generate an executable that can be flashed on the respective ECU hardware.

«Requirement»
Req5.1-RTEGeneration_Configuration
ID = Req5.1
The tool must take all the intermmediate artefacts as inputs and generate RTE.

«Requirement»
Req5.4-RTECodeGeneration
ID = Req5.4
The tool must provide editors and interfaces to generate AUTOSAR RTE C-Code.

«Requirement»
Req5.2-Language Compatibility
ID = Req5.2
The tool provides C/C++ programming interfaces with the respective compilers.

«Requirement»
Req5.5-ComponentTypes
ID = Req5.5
The tool provides necessary interfaces to the application component types and RTE.

«Requirement»
Req5.9-ExclusiveAreas_SystemPartitioning
ID = Req5.9
The tool must be able to handle exclusive areas and system partitioning.

«Requirement»
Req5.6-Commuincation Interfaces
ID = Req5.6
The tool provides communication interfaces and system signal communication.

«Requirement»
Req5.8-MappingRunnablestoOS
ID = Req5.8
The RTE configurator tool provides interfaces to map runnabels in execution order to OSTasks.

«Requirement»
Req5.11-Signals_TimingAnalysis
ID = Req5.11
The tool must provide interfaces to map the data to signals and also provide interfaces to analyze worst case execution time and worst case run time.

«Requirement»
Req5.7-ImplementationTypes_CalibrationParameters
ID = Req5.7
The tool also provides implementation types and calibration parameters.

«Requirement»
Req5.10-RTEEventstoOSEvents
ID = Req5.9
The tool must generate connections to the RTE Events to OS Events.

«Requirement»
Req5.3-VerificationValidation
ID = Req5.3
The tool should be able to verify the real-time behavior of the system and ensures reliability.

Figure 4.7: Stakeholder requirements diagram for RTE Tools

<trace> requirement relationship provides a general-purpose relationship between a requirement and any other model element, as shown in Figure 4.4. The <trace> relationship here is being used for relating requirements to source documentation or for establishing a relationship between main specifications.

With this, **RQ2.2** is answered.

4.4 Architectural modeling

The architecture of a system is the set of fundamental concepts or properties of that system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [74]. It serves as a blueprint of the overall system behavior. The significance of an architecture is to aid in communication, analysis and construction of the system at hand. A system has multiple stakeholders who have different concerns, requirements, constraints and expect different outcomes from the system under development. A good architecture is one that successfully addresses the concerns of its stakeholders and, when those concerns are in conflict, balances them in a way that is acceptable to the stakeholders.
An architecture is described by a collection of models. Modeling plays a vital role in the development of an automotive ECUs, as it provides a higher abstraction level, thereby reducing the complexity. In this case, models are used to give a better understanding of how each stakeholder interacts with the system. As such, models are used for understanding, analysis, communication, construction, documentation and for answering other questions based on the system (for eg. cost and risk, evaluation of utility etc.). The models here depict logical viewpoint of all the stakeholders, who are the users of AUTOSAR tool-chain. A logical view focuses on the functionalities imparted by the system (AUTOSAR tool-chain) to its end users.

4.4.1 Model tools

AUTOSAR modeling tools are mainly used to perform architecture design, requirements management, network communication design, application design and wiring harness design. They facilitate early testing via VFB. Using the model tools, an optimal configuration for the application use-case can be put together. A Requirement Analyst and a Software Designer works together in this case (normally such a tool can be used by architects, network designers, development engineers, test engineers, but in this case the scope is minimized for the above mentioned stakeholders to reduce the complexity in roles).
The main goals of the software designer is to model the ASW and network architecture for an ECU to be developed. Together with the requirement analyst, most important functionalities which are desired from an AUTOSAR modeling tool are analyzed using the use-case diagram. Here, only the most important requirements are considered as per the AUTOSAR methodology.
The main aim of AUTOSAR is to co-operate on standard but compete on implementation. In this regard, the tool-vendors also incorporate multiple extra tool features in order to increase the value of their tool-chain. These extra features are considered while the specific tools are ranked in Section 4.6.

4.4.2 Application tools

The application developer generates C code for the respective application. While the modeling tools are used to model all the aspects of an application such as ports, interfaces, network connection, internal behavior etc., ASW code generation tool is used to implement the core functionality of runnables.

4.4.3 Basic software tools

The system designer and system developer have the task of generating the BSW code and configuring ECU modules. Accordingly sub requirements (desired tool functionality) are derived and are traceable to their respective main requirements.

4.4.4 RTE tools

Similarly, RTE tool model has the main task of generating the RTE layer performed by system engineer. This tool can also be used to further test and validate models.
All use case diagrams can be found in Appendix A1. This completes answering the research question **RQ3.1**.

4.5 Criteria selection

The selection criteria is a list of essential and desired tool attributes which is necessary for successfully implementing AUTOSAR architecture. All the functional and non-functional aspects must be considered in this regard. At first, 6 top-level criteria are considered which drives the decision driver models. They are *functionality*, *usability*, *interoperability*, *service and support*, *cost and distribution* and *testability.* Out of them, functionality, interoperability and testability are considered the most essential criteria (also while assigning weights) while some compromises could be made with the remaining ones.

- Functionality
  Functionality is a metric to qualify if a tool alternative comprises of the most essential requirements described in the Figure 4.3.

- Usability
  Usability takes into account the stakeholder requirements like ease of tool usage, graphical user interface and an overall user experience.

- Interoperability
  An interoperable system makes it easier to exchange and reuse information both with the tool-chain and with a different tool-chain which implies that the selected tool-chain must be compliant in terms of specific automotive standards (eg., OSEK, MISRA), bus protocols (eg., CAN, LIN) and data exchange formats (eg., .ARXML) adopted within the tool-chain. Further, it also supports integration with third party tool-chain.

- Service and support
  Service and support refers to the after sales service provided by tool vendors imparting the knowledge and best practices of using the tool, support in terms of proprietary software migration to AUTOSAR etc.

- Cost and distribution
  Cost and distribution refers to the initial and ongoing investments (cost of entire tool-suite), types of licenses, distribution cost, time taken to deliver the tool, in what form (downloadable, dongle) and other considerations that cost money are taken into account.

- Testability
  Testability refers to the ability of the tools to support early stage verification of the application and BSW models.

Each of these top-level criteria are further divided into sub-criteria based on specific qualifying parameters. The use case diagrams and requirement diagrams along with the AUTOSAR knowledge gathered are used, to derive specific criteria for selecting the tool-chain, as shown below:

1. Functionality

- System modeling
- Modeling analysis
- Network modeling
- System configuration
- Code generation
- Timing analysis

2. Usability
   - High-level GUI (ease-of-use)
   - Re-usability
   - Modifiability
   - Intuitiveness
   - Documentation

3. Interoperability
   - Data exchange
   - Compliance to standard
   - AUTOSAR Interfaces
   - Standard protocols and libraries
   - Integration with third-party tools

4. Service and support
   - Migration support
   - Workshops / webinars (imparting tool knowledge)

5. Cost and distribution
   - Tool-suite cost (license)
   - Duration of license
   - Format and distribution time
   - Market penetration
   - Latest release
   - Release interval

6. Testability
   - Debugging
   - Simulation
   - Virtual ECU platform for early testing and verification (Virtual Function Bus)

Hence, research questions **RQ3.2, 3.3** are answered.

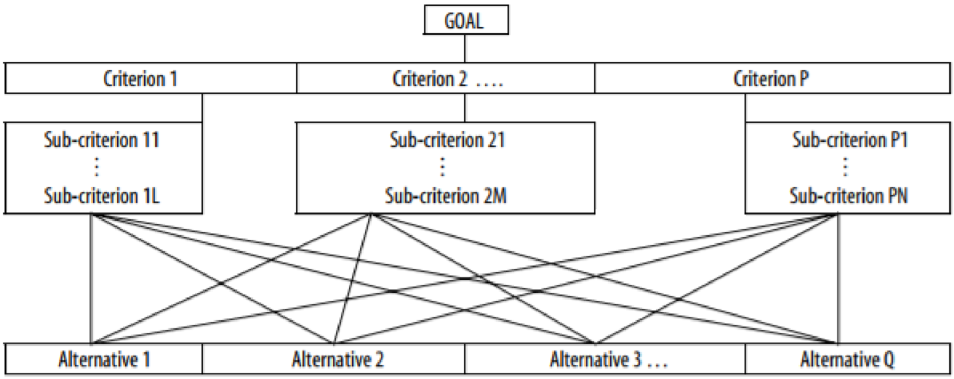


Figure 4.8: Analytic Hierarchy Process

4.6 Analytic Hierarchy Process (AHP)

This section describes the therory behind the AHP algorithm in detail which is further applied in the next section. AHP uses a mathematical approach based on metrics algebra [30]. The tool is used to prioritize each criteria in decision making in order to achieve a certain goal. AHP applies a qualitative approach to restructure problems into a hierarchy which is more systematic. On the other hand, based on a quantitative approach, it uses a pair-wise comparison method to select alternatives for a particular criteria that are more consistent. Fundamentally, AHP algorithm operates by prioritizing competing alternatives as well as the criteria used to judge these alternatives. AHP includes 6 main steps. They are:

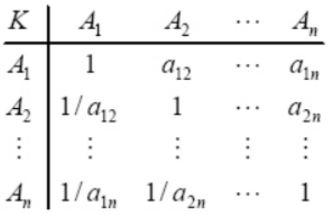
1. Identify goals and criteria and represent them in a hierarchical framework.
   Before implementing the AHP algorithm, it is important to understand the nature of a problem and collate a list of requirements that are pertinent to the problem. This helps in identification of the main goal, sub-goals and criteria. Further, the problem is decomposed into a hierarchy of goals, criteria, sub-criteria and alternatives. A hierarchy decomposes a problem and indicates a relationship between elements of level with those of the level immediately below. This is the creative part of decision making which results in an ordered network. Saaty [29] suggests that a useful way to structure the hierarchy is to work down from the goal as far as one can and then work up from the alternatives until the levels of the two processes are linked in such a way as to make comparisons possible. Figure 4.4 shows a generic hierarchical structure. Root node of a hierarchy is the goal which signifies the objective of the problem being solved, which is followed by the sub-nodes called criteria and sub-criteria. It is important to note that when comparing elements at each level a decision maker has to compare with respect to the contribution of the lower level elements to the upper-level ones. This local concentration of the decision-maker on a single part of the whole problem is a powerful feature of AHP.

2. Perform pair-wise matrix comparisons
   Next step is to construct pair-wise comparison matrix and to establish priorities by comparing each criterion within the hierarchy. A pair-wise comparison method is used to determine the importance of each upper level node w.r.t the immediate lower level node, in this way the decision maker quantifies the relative importance of each single pair-wise comparison at each comparison. For this, Saaty proposed a qualitative scale as shown in Table 4.1. Experts can rate the comparisons as equal, marginally strong, very strong and extremely strong. For example, A is more important than B, B is of the same importance as D, C is less important than D etc. Accordingly, each decision is quantified with a value between 1 and 9 as shown in the Table 4.1. If a choice cannot be made within the available choices, for example, if an opinion falls in between two intermediate judgments i.e. equal importance and moderately

equal importance, then a value 2 could be chosen and so on. In a matrix of i rows and j columns, a comparison is made between each alternative w.r.t. criteria K as shown in the Figure 4.4. If an alternative A_{ij} has one of the non-zero numbers (shown in Table 4.1) when compared with j, then A_{ji} gets the reciprocal value when compared with i.

Table 4.1: Pairwise Comparison Table

Intensity of Importance	Definition	Explanation
1	Equal importance	Two elements contribute equally to the objective
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Strong importance	Experience and judgment strongly favor one element over another
7	Very strong importance	One element is favored very strongly over another, its dominance is demonstrated in practice
9	Extreme importance	The evidence favoring one element over another is of the highest possible order of affirmation



K	A_1	A_2	$\cdots$	A_n
A_1	1	a_{12}	$\cdots$	a_{1n}
A_2	$1/a_{12}$	1	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_n	$1/a_{1n}$	$1/a_{2n}$	$\cdots$	1

Figure 4.9: Pair-Wise Comparison Matrix

3. Extract the relative importance of previous comparisons (weights)
After having obtained all the pair-wise inputs from the subject matter expert, normalize the matrix to ensure consistency of values. For a matrix of Pair-wise elements:

$$C_{ij} \implies \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

Summation of each column of pair-wise matrix:

$$Sum_j = \sum_{i=1}^{n} C_{ij}, \text{ where,}$$
$$Sum_j = \text{summation of each column for each } j = 1 \text{ to n}$$

Divide each element in the matrix by its column total to generate a normalized pair-wise matrix.

$$X_{ij} \implies \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} = \frac{C_{ij}}{Sum_j}$$

Further, the principle eigenvalue and the corresponding normalized right eigenvector of the comparison matrix gives the relative importance of the various criteria being compared. The elements of the normalized eigenvector are called weights w.r.t the criteria or the sub-criteria and comparisons w.r.t the alternatives. This is done by computing the geometric mean of each row.

$$\vec{W_i} \implies \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} = \frac{\sum_{j=1}^{n} X_{ij}}{n}, \text{ where,}$$

$$\vec{W_i} = \text{priority vector or weighted matrix of each row } i$$

4. Perform consistency analysis
The consistency of the matrix of order *'n'* is given by calculating the Consistency Ratio (CR). At first, the consistency vector ($\vec{C_i}$)is calculated by computing the product of pair-wise matrix (C_{ij}) with the priority vector ($\vec{W_i}$). i.e,

$$\vec{C_i} \implies \begin{bmatrix} \vec{C_1} \\ \vec{C_2} \\ \vec{C_3} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} * \begin{bmatrix} \vec{W_1} \\ \vec{W_2} \\ \vec{W_3} \end{bmatrix}, \text{ where,}$$

$$\vec{C_i} = \text{consistency vector}$$

The next step is to calculate maximum eigenvalue (λ_{max}), which is further required to compute the values of Consistency Index (CI) subsequently.

$$\lambda_{max} = max(\frac{\sum_{i=1}^{n}(\frac{\vec{C_i}}{\vec{W_i}})}{n}), \text{ where,}$$

$$\lambda_{max} \text{ is the maximum eigenvalue of the pair-wise matrix } C_{ij}$$

$$CI = \frac{\lambda_{max}-n}{n-1}$$

The final step is to calculate the value of CR, which is given by the formula:

$$CR = \frac{CI}{RI} < 10\%, \text{ where,}$$

RI is the index of a *randomly* generated pair-wise comparison matrix

The value of RI depends on the number of items being compared and specific RI values are provided by Saaty based on the order of random matrix (*'n'*) as shown in the Table 4.3.

Table 4.3: Random Index of consistency for corresponding order of matrix

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

The upper row indicates an order of random matrix while the lower row indicates the corresponding values of consistency for random judgments. These values are derived by averaging CIs from a sample of randomly selected reciprocal matrices of AHP method. Saaty suggests the value of CR should be less than 10%, which implies that the adjustment is small compared to the actual values of the eigenvector entries. If the value of CR fails to reach the required level (say 90%), then it indicates that the pair-wise judgments are just about random and cannot be trusted. Therefore, in this case the values entered in matrix C_{ij} has to be re-examined.

4.7 Application of Analytic Hierarchy Process(AHP)

In this section, the AHP algorithm which is described above, is applied to the set of criteria and sub-criteria with the goal to make a selection for AUTOSAR tool-chain. At first, the goal is explained in detail and a hierarchical structure is presented. Further, pairwise comparisons are made between each criteria and finally, the tools are compared based on the selection metrics.

4.7.1 Goal identification and representation of goals and criteria in a hierarchical framework

The main goal was to rank and select an AUTOSAR tool-chain for two different key selection criteria from the available tool alternatives. These two criteria were:

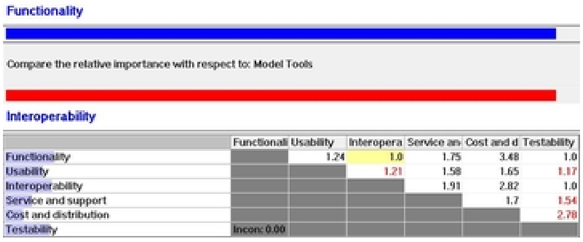
1. Best *performing* tool

2. Best *low-cost* tool (cost is the main criteria while other aspects gets less score)

Each of the these goals has sub-goals which are, selection of Model tool, ASW tool, BSW tool and RTE tool, which together constitute a tool-chain. To each of these sub-goals, AHP is applied considering the criteria and sub-criteria mentioned in Section 4.5. A generic hierarchical framework is as shown in Figure 4.10. The main goal is divided into 4 sub-goals (corresponding with the four categories of tools), each of which has 6 selection criteria. Each criteria has multiple sub-criteria. A pairwise comparison is made between each of these criteria in order to select an alternative, using the Expert Choice AHP tool.

4.7.2 Pair-wise matrix comparisons

In this step, at first, each criteria (e.g. functionality, interoperability etc.,) are weighed within each sub-goal to determine the hierarchy of importance for each of these criteria and to what extent can they be compromised without affecting an overall performance of the tool. Figures 4.4 to 4.7 shows pair-wise matrix comparisons for all four tool categories. If any two criteria are equally important, then they get a score of 1.0 (e.g. boxes highlighted in yellow, i.e. both the bands (red and blue) corresponding to Interoperability and Functionality are dragged on the same level to indicate an equal importance for both criteria).

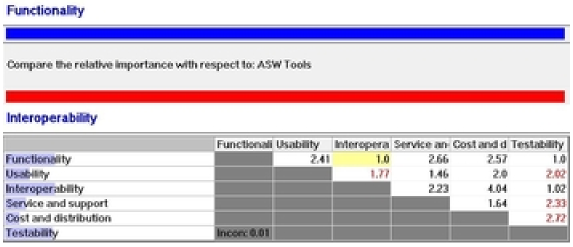

Functionality

Compare the relative importance with respect to: Model Tools

Interoperability

	Functionali	Usability	Interopera	Service an	Cost and d	Testability
Functionality		1.24	1.0	1.75	3.48	1.0
Usability			1.21	1.58	1.65	1.17
Interoperability				1.91	2.82	1.0
Service and support					1.7	1.54
Cost and distribution						2.78
Testability	Incon: 0.00					

Figure 4.10: Model tools - Pair-wise comparisons

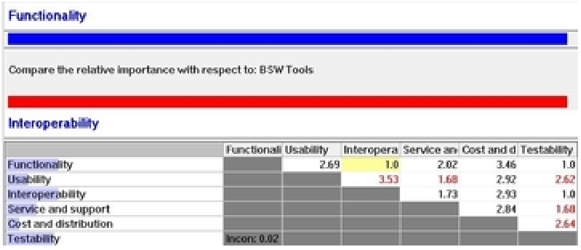

Functionality

Compare the relative importance with respect to: ASW Tools

Interoperability

	Functionali	Usability	Interopera	Service an	Cost and d	Testability
Functionality		2.41	1.0	2.66	2.57	1.0
Usability			1.77	1.46	2.0	2.02
Interoperability				2.23	4.04	1.02
Service and support					1.64	2.33
Cost and distribution						2.72
Testability	Incon: 0.01					

Figure 4.11: ASW tools pair-wise comparisons

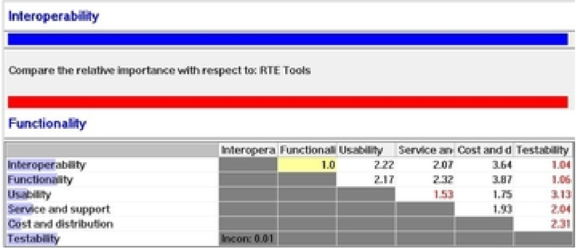

Functionality

Compare the relative importance with respect to: BSW Tools

Interoperability

	Functionali	Usability	Interopera	Service an	Cost and d	Testability
Functionality		2.69	1.0	2.02	3.46	1.0
Usability			3.53	1.68	2.92	2.62
Interoperability				1.73	2.93	1.0
Service and support					2.84	1.68
Cost and distribution						2.64
Testability	Incon: 0.02					

Figure 4.12: BSW tools pair-wise comparisons


Interoperability

Compare the relative importance with respect to: RTE Tools

Functionality

	Interopera	Functionali	Usability	Service an	Cost and d	Testability
Interoperability		1.0	2.22	2.07	3.64	1.04
Functionality			2.17	2.32	3.87	1.06
Usability				1.53	1.75	3.13
Service and support					1.93	2.04
Cost and distribution						2.31
Testability	Incon: 0.01					

Figure 4.13: RTE tools pair-wise comparisons

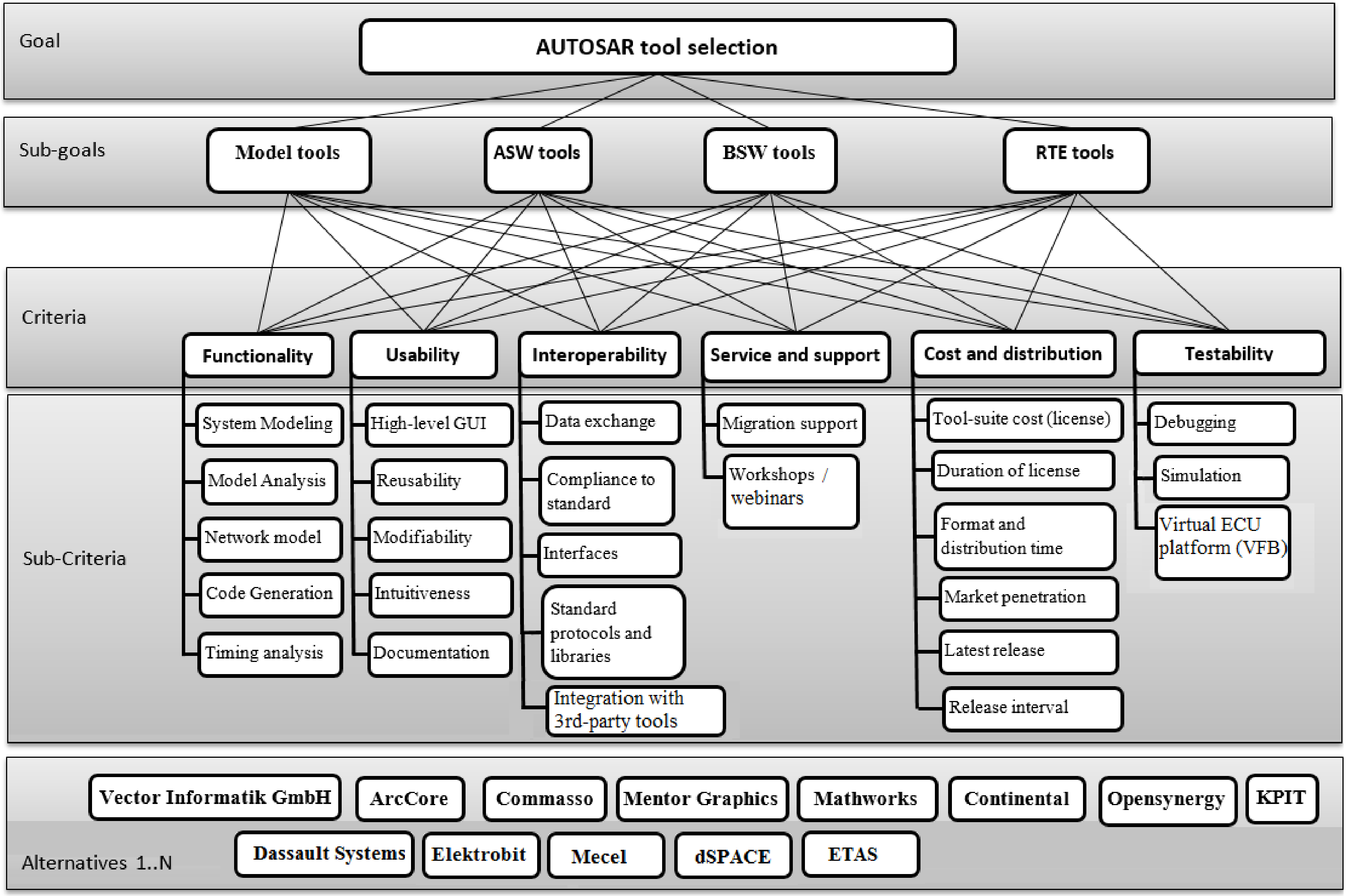


Figure 4.14: AHP hierarchical framework of goals and criteria

Other specific values in the table are derived by dragging the bands according to the specific importance of their respective criteria for each sub-goal and thus a pairwise matrix is generated. Independent criterion are not compared with each other (e.g. system modeling, high-level GUI etc.,) since all of them are the attributes of a good tool and one cannot be graded over another. The corresponding priority graphs for all four pair-wise comparison matrices above, are as shown in Figures 4.11 to 4.14, respectively (alternative representation of Figures 4.6 to 4.9). These graphs are sorted based on the priority ranking for each criteria of each sub-goal (weights).

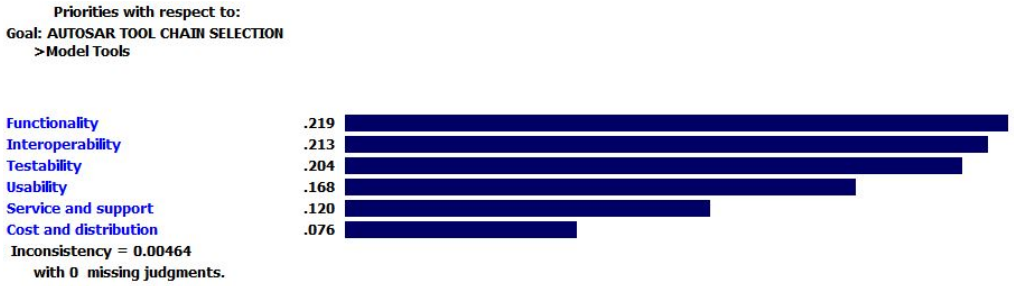


Figure 4.15: Priority graph w.r.t. Model tools

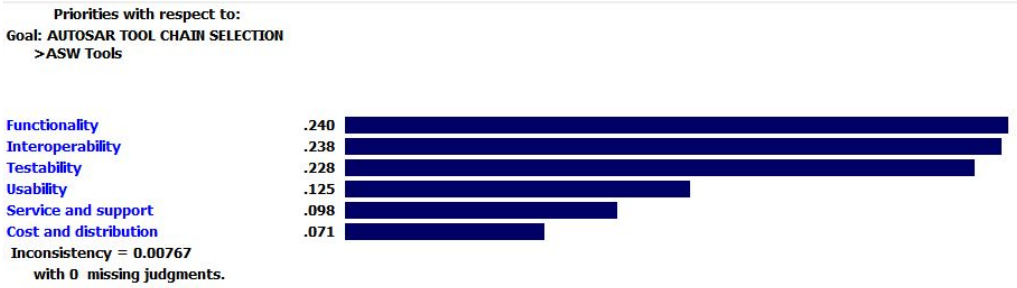


Figure 4.16: Priority graph w.r.t. ASW tools

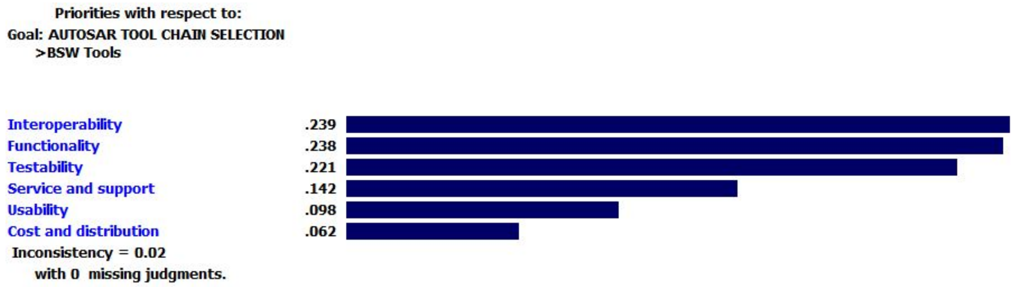


Figure 4.17: Priority graph w.r.t. BSW tools

4.7.3 Synthesis (relative importance) and Consistency ratio (CR)

Consistency Ratio determines an overall inconsistency of all the weights within a pairwise matrix (for example, if Tool A is weighted greater than Tool B and B is weighted greater than Tool C, then A should be weighted greater than C). According to AHP algorithm, the value of CR should be less than 0.1, in order to determine that the weights and calculations are consistent. In Figure 4.15, synthesis w.r.t the main goal is performed and an overall CR is generated, i.e. 0.01 and hence

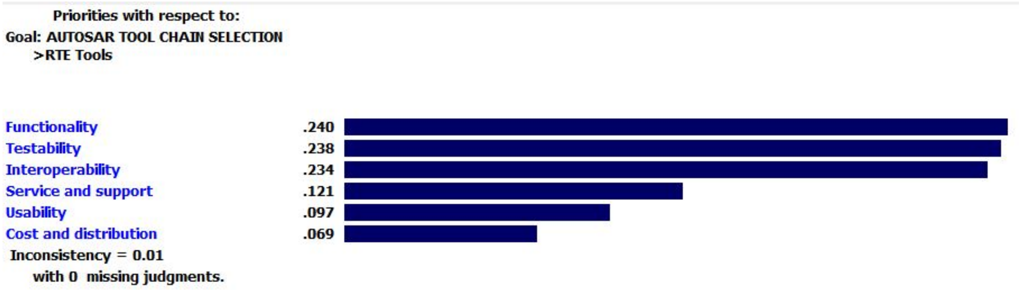


Figure 4.18: Priority graph w.r.t. RTE tools

the weighted values are consistent. In order to avoid any rank reversal between the alternatives, *ideal mode* was used while calculating CR.

All pairwise comparison matrix and synthesis graphs for each criteria and sub-goals can be found under Appendix A3.

In order to rank the 13 tool alternatives within each criteria under each sub-goal, as shown in Figure 4.10, following procedure was followed:

1. Step 1
   Section 3.5 of Chapter 3 gives a brief overview on various features provided by each software tool vendor. Tables 3.1 and 3.3 further summarizes the capacity of features within each tool-chain. This enables us to understand what aspect of the AUTOSAR methodology is covered by a tool-chain and what is lacking. Note: MCAL tool feature is not explicitly stated since its a part of BSW layer.

2. Step 2
   Next, the sub-criteria listed in Figure 4.10 are considered and the tools are evaluated to see if they conform to all the required attributes. The evaluation of a particular tool (e.g. Vector Informatik Gmbh) is always in comparison with another tool (e.g. ArcCore).
   For example, under the criteria interoperability for model tools (A3 .2 Figure 20), all AUTOSAR modeling tools provided by the tool vendors facilitate the creation of ECU Extract in the standard data exchange format (.ARXML). Further, all tools abide to the standardized features of AUTOSAR architecture thereby providing standard interfaces, protocols and libraries. Hence, all the tools under this criteria are graded equal.

3. Step 3
   While comparing any two alternatives w.r.t. a particular criteria and sub-goal (e.g. comparison of *Vector* and *ArcCore* w.r.t. *Functionality* of *Model Tools*), the main question answered is, which alternative is preferred w.r.t. that criterion for that sub-goal. After performing all the comparisons for a sub-goal, general preference of alternatives is calculated as a weighted sum of the criteria's priorities (Figures 4.6 to 4.9) along with alternative's priorities (see appendix A3). At this point, it is also important to ensure that the inconsistency values for all the comparisons are less than 0.1. Accordingly, all other criteria for each sub-goal are evaluated and pairwise comparison matrix is generated.
   Note: If a particular tool-chain lacks specific features under any of the sub-goal, its overall rank will be low (e.g. dSPACE does not provide any tool features necessary for modeling or application development. Eventhough its ranked high under BSW tools and RTE tools (see Appendix A3), its overall rank is comparatively low (Figure 4.15)).

4. Step 4
   After comparing tools for all the criteria and sub-goals, next step is to synthesize these results (Figure 4.15) and finally generate graphs, which is further explained in next section.

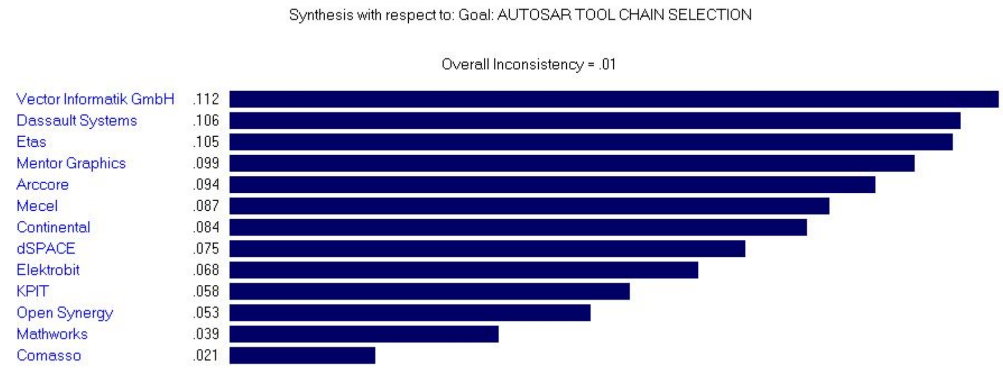


Figure 4.19: Synthesis graph for all the tool alternatives

4.7.4 Overall priority ranking and tool selection

An overall priority is determined for all sub-goals and criteria, i.e. weights are assigned to rank all sub-goals and criteria as shown in graphs (Figure 4.15, Appendix A.3). BSW tools and RTE tools are ranked / prioritized slightly higher than Model tools and ASW tools. The reason is because configuring BSW and RTE are the most complex parts in implementing AUTOSAR architecture. Though modeling and application generation is equally important, some compromises can be made when compared to BSW and RTE tools, mainly with regards to complexity in applying BSW and RTE tools. Further, the AUTOSAR tool from Vector Infomatik GmbH performed better overall. AHP results for individual tool categories can be found under Appendix A.2.

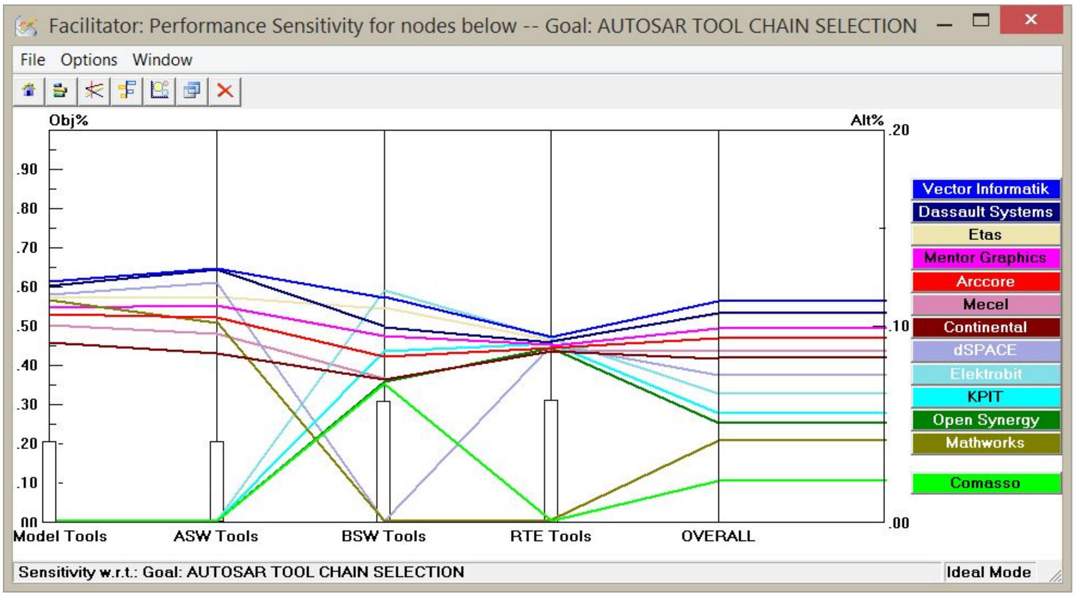


Figure 4.20: Overall tool ranking

Finally, a tool-chain is selected each for performance metric and cost metric. This method of comparing the alternatives gives a better overall perspective of the efficiency of a tool in doing a certain task i.e., a tool-vendor can have 1 tool performing both modeling and generating ASW code. But to what extent can it perform better than the other can be assessed using this methodology. With this approach, if a certain tool-vendor provides 3rd party integration capabilities for the tools, then instead of buying a whole tool-suite from just one vendor, it might prove cost-effective for a small company to go for AUTOSAR tooling options from more than one vendor that satisfies the requirements with some trade-offs made. But if cost is not a main bottleneck, then this methodology still proves efficient as it provides a clear idea of the strengths of the tool-chain and if that matches the desired requirements.

4.8 Tool selection

After applying AHP to AUTOSAR tool-chain the tools selected for best overall performance are AUTOSAR tool-chain from Vector Informatik GmbH, Dassault Systems, ETAS, Mentor Graphics and ArcCore (in that order). The low-cost AUTOSAR tools selected are Mathworks Embedded Coder for modeling tool, ArcCore for procuring only the Arctic Studio and COMASSO community membership for the procuring the BSW modules. Matlab Embedded Coder was mainly selected due to its rich GUI features. If having a GUI is not an essential requirement and if compromises can be made in terms of usability, then Arctic Studio can be used also as a low-cost modeling tool. Therefore, **RQ4.1, 4.2** are answered.
In this thesis work, only ArcCore's Arctic Studio and Arctic Core was used in implementing AUTOSAR architecture and for evaluation of the tool, because of the availability of trial license.

4.9 Trade-offs

While selecting AUTOSAR tool-chain prioritizing cost as the key selection criteria, certain trade-offs were considered to be made. They are as follows.

1. Least priority was given to *usability*, *service and support* and other value added services that costs extra charges. Instead required knowledge can be derived from available (free) resources.

2. Some features for testing like efficient debuggers, simulations are compromised. Instead any third party tools for simulation and testing could be used (if available, not researched in this work).

3. The files exchanged between the tools (.ARXML) might not be seamlessly *interoperable* and compromises had to be made in this regard. This might imply that some changes might be made within the tool to integrate with other tools, but this could be a complex task prone to errors. Hence, not recommended.

Keeping all the above trade-offs in mind, weights for attributes *functionality, interoperability and testability* are not significantly reduced even-though *cost* is given the highest priority (graphs can be found in Appendix A.2 and A.3). Thus research question **RQ4.3** is answered.

Chapter 5

Demonstrator

As a part of the case study for this work, a demonstrator using the AUTOSAR architecture was implemented using the selected AUTOSAR tool. This demonstrator was used as a means to apply the tool selected in Chapter 4 i.e. Arctic Studio (Release 15.x.x), based on previously selected criteria. This demonstrator also helps in understanding the prospects and challenges of using low-cost methods in implementing the AUTOSAR architecture. In the following sections, the system design for the demonstrator is explained in two phases. The initial design was a failed attempt due to low-cost constraints of the project and novice understanding of the tool-suite. The many roadblocks and challenges during the initial design phase led to a better understanding and usage of tools, as a result, an updated design is presented subsequently. Final section summarizes the tools required for an implementation of the demonstrator.

5.1 Initial design and challenges faced

5.1.1 System requirements

Initially, stakeholders were identified and a set of requirements were put forth before designing the demonstrator. Main stakeholders of this project were, Orlaco, as a client that provided EMOS camera for this project and BRACE Automotive which was the owner of the end product. Other stakeholders included project manager and project developer, performed by me under the guidance of BRACE Automotive. The stakeholder requirements that were put forth were as follows:

1. Use of low-cost tools and off-the shelf components
   Since the main emphasis of this thesis was to explore the possibilities of using the AUTOSAR architecture in the realm of a limited budget and resources, there was a high emphasis on making use of off-the-shelf hardware components along with the selected low-cost tool. In this regard, Arduino Mega 2560 board was used to implement AUTOSAR. This was supplemented with Arduino's CAN and Ethernet shield for implementing CAN and Ethernet communication.

2. Use of Arctic Studio tool-suite for implementing AUTOSAR
   Though the entire ArcCore's tool solutions were not considered low-cost (as illustrated in Chapter 4), Arctic Studio along with Arctic Core (release 15.x.x) were made use of due to their availability and ease of access. However, ArcCore's Arctic Core did not provide support for the chosen microcontroller for this project. Hence, most part of the basic software needed to be implemented manually.

3. Manual implementation of the MCAL layer
   Since the only available off-the-shelf component for implementing AUTOSAR was Arduino Mega 2560, it was decided to be used as an underlying hardware platform for the demonstrator. But during discussions with ArcCore it was found that ArcCore did not support

the hardware drivers for the ATMega micro-controller family. Hence, it was decided that the required drivers were to be implemented manually.

4. Manual implementation of compiler-scripts
   Though Arctic Studio supports multiple compilers, it does not have support for avr-gcc compiler. Hence, it was also required to put the avr-gcc compiler script and other libraries in place in order to support the micro-controller that was being used.

5. Manual implementation of Operating System wrapper
   It was also decided that the operating system was going to be implemented manually or a wrapper around the existing OS is coded which is compatible with Arduino microcontroller.

6. Implementing CAN and Ethernet communication stack
   As EMOS camera was an Ethernet device, the idea was to establish a communication between the Ethernet (EMOS Camera) and the CAN communication channels (CAN Simulator) using AUTOSAR architecture which was implemented on Arduino. The motive was to keep the application realistic yet simple.

7. Configuring only the required BSW modules
   The BSW layer of AUTOSAR consists of over 63 modules which forms the bulk of an entire architecture. In order to reduce the complexity only those required modules for implementing the demonstrator were adopted.

8. Simple application yet realistic
   Number of software components were considered to be atleast 3 or more with more than one type of SWCs (Application SWC, Sensor/Actuator SWC, etc.). Further, it should implement both interfaces (client server, sender receiver). Communication with other SWCs and underlying BSW layer had to be established. Accessing I/O interfaces was also a necessary requirement.

Accordingly a conceptual design was made as shown in Figure 5.1. The topology diagram shown in Figure 5.2, shows the basic set up of the demonstrator being developed in a real life scenario.

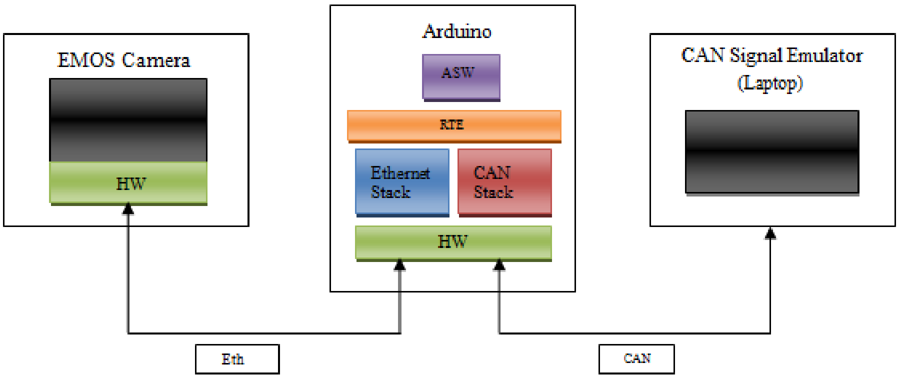


Figure 5.1: Initial design of the demonstrator

5.1.2 System hardware

The hardware components that were used to implement the demonstrator are briefly explained below:

1. Arduino Mega 2560
   Arduino Mega 2560 is an 8-bit Atmel microcontroller with 8KB SRAM and 256KB flash

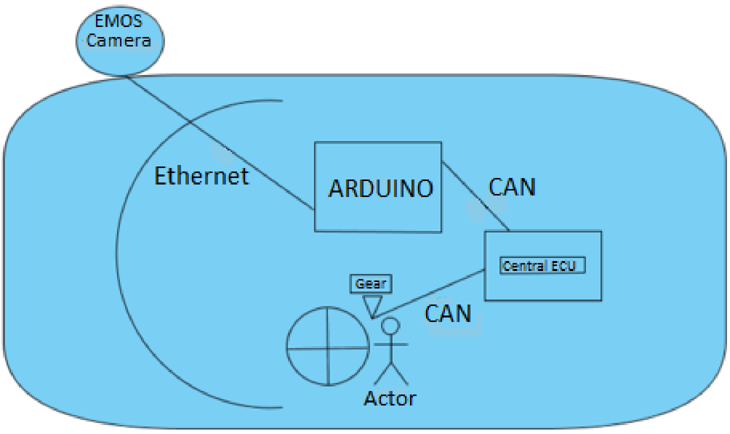


Figure 5.2: Topology diagram for the demonstrator

memory [76]. It is also compatible with Arduino CAN and Ethernet shield which is used to establish Ethernet and CAN communication.

2. EMOS Camera
EMOS camera is developed in-house by Orlaco B.V and is mainly used in heavy duty vehicles like trucks, cranes, mining equipment etc [77]. It is connected to a monitor via an Ethernet cable and streams images (MJPEG) using RTP stream protocol. The Real-time Transport Protocol (RTP) is a network protocol for delivering audio and video over IP networks. Further, it features a latency less than 100ms.

3. PEAK CAN adapter
The PEAK CAN adapter is an USB device which enables simple CAN communication within the network. This adapter is used to emulate the CAN signal on a laptop (PCAN View) through which CAN messages can be sent to the Arduino board.

4. Lenovo laptop
The entire implementation was done using a Lenovo thinkpad which runs on Windows operating system.

5.1.3 System software design

As mentioned in the requirements, the application was intended to be simple yet realistic. Accordingly, three user stories were put forth, with regards to a driver of a truck being the user, of the demonstrator being implemented.

1. User story 1
As a driver, it is intended that the demonstrator automatically changes the camera display from front view to rear view by sensing the change in transmission on a reverse gear, so that manual interaction with the system can be minimized.

2. User story 2
As a driver, it is intended that the demonstrator changes the Region Of Interest (ROI) of the camera on cross-roads, so that blind spots are easily visible.

3. User story 3
As a driver, it is intended that the demonstrator displays an indicator or an LED light (either green or red) indicating the status of the data stream, so that the driver can be certain if the stream is being transferred or if there has been some problem. If then the driver can view the real mirror instead of the camera display.

Functional Requirements

As per the above mentioned user stories, following are the functional requirements for the demonstrator:

1. The Arduino ECU which implements AUTOSAR architecture, must provide CAN and Ethernet communication interfaces to send and receive CAN and Ethernet messages.
2. Arduino should receive *transmission status* and *steering angle* as inputs via CAN in order to process the data further.
3. In order to satisfy User Story 1, Arduino should monitor the change in transmission signal received over CAN, periodically. If the transmission signal received is on *reverse gear*, then it should send a signal to EMOS camera to change its view from 'front' to 'rear' view over the Ethernet communication channel.
4. In order to satisfy User Story 2, upon receiving the steering angle via CAN, Arduino should send an appropriate ROI value over Ethernet to the EMOS camera. By altering the value of ROI, the display can be zoomed thereby detecting any blind spots which the driver can see clearly.
5. In order to satisfy User Story 3, Arduino should access an on-board LED device and blink periodically if the data stream has been detected. If the data stream is not detected over EMOS camera, LED remains switched off.

Application design

Accordingly, application SWCs were designed and connected as shown in Figure 5.3 and 5.4.

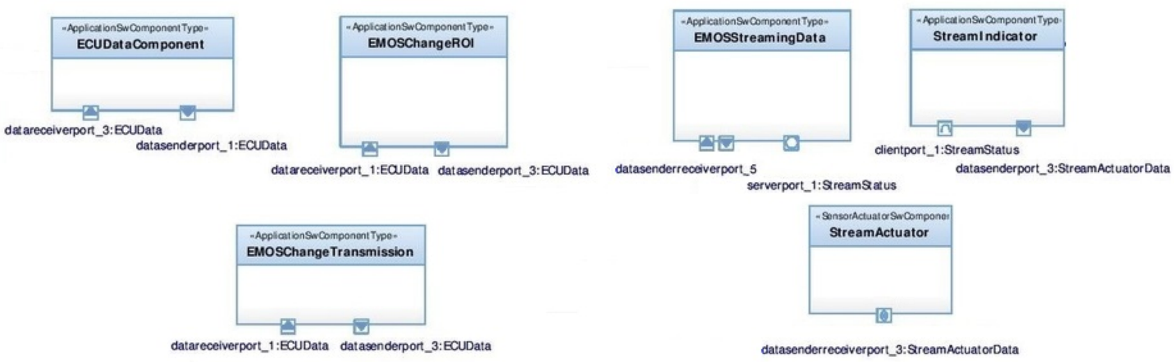

«ApplicationSwComponentType»
ECUDataComponent
datareceiverport_3:ECUData
datasenderport_1:ECUData
«ApplicationSwComponentType»
EMOSChangeROI
datareceiverport_1:ECUData
datasenderport_3:ECUData
«ApplicationSwComponentType»
EMOSStreamingData
datasenderreceiverport_5
serverport_1:StreamStatus
«ApplicationSwComponentType»
StreamIndicator
clientport_1:StreamStatus
datasenderport_3:StreamActuatorData
«ApplicationSwComponentType»
EMOSChangeTransmission
datareceiverport_1:ECUData
datasenderport_3:ECUData
«SensorActuatorSwComponen
StreamActuator
datasenderreceiverport_3:StreamActuatorData

Figure 5.3: Software Components

The connections could be summarized as follows:

Table 5.1: Application design

Sender/Client SWCs	Receiver/ Server SWCs	RTE Events	Interface	Data Elements
EMOSDataComponent	EMOSChange ROI	dataWriteAccess/dataReadAccess	S/R	Steering angle
	EMOSChange Transmission	dataWriteAccess/dataReadAccess	S/R	Transmission value

StreamIndicator	EMOSStreamingData	operationInvoked Event/serverCallPoint	C/S	Stream status
	StreamActuator	dataWriteAccess/dataReadAccess	S/R	Stream status

Further, the ports that are open in Figure 5.4, receive or send signals to the underlying layers, which is summarized in Table 5.3.

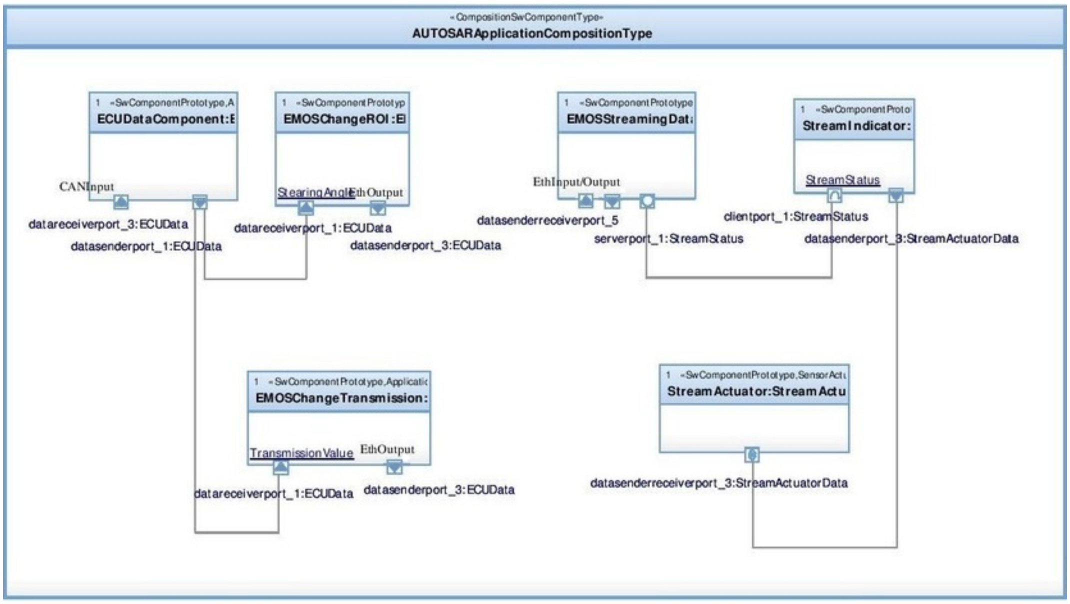

«CompositionSwComponentType»
AUTOSARApplicationCompositionType
1 «SwComponentPrototype,A
ECUDataComponent:E
1 «SwComponentPrototyp
EMOSChangeROI :E
1 «SwComponentPrototype
EMOSStreamingDat:
1 «SwComponentProto
StreamIndicator:
CANInput
StearingAngle
EthOutput
EthInput/Output
StreamStatus
datareceiverport_3:ECUData
datasenderport_1:ECUData
datareceiverport_1:ECUData
datasenderport_3:ECUData
datasenderreceiverport_5
serverport_1:StreamStatus
clientport_1:StreamStatus
datasenderport_3:StreamActuatorData
1 «SwComponentPrototype,Applicatic
EMOSChangeTransmission:
TransmissionValue
EthOutput
datareceiverport_1:ECUData
datasenderport_3:ECUData
1 «SwComponentPrototype,SensorActu
StreamActuator:StreamActu
datasenderreceiverport_3:StreamActuatorData

Figure 5.4: Application design

Table 5.3: Communication matrix

I-Pdu	SWCs	RTE Events	Interface	Data Elements
Rx CAN Signal	ECUDataComponent	dataWriteAccess/dataReadAccess	S/R	ECUData
Tx Eth Signal	EMOSChangeROI	dataWriteAccess/dataReadAccess	S/R	ROI value
	EMOSChangeTransmission	dataWriteAccess/dataReadAccess	S/R	Change view status
Tx / Rx Eth Signal	EMOSStreaming Data	timingEvent	S/R	Stream status

In Figure 5.4, ECUDataComponent receives two CAN data signals from the underlying COM module (Figure 5.5). It sends the received data to EMOSChangeROI and EMOSChangeTransmission SWCs which on further processing, sends the data required i.e., *ROI value* and *change view status (Y/N)*, to EMOS camera via Ethernet. Client application SWC StreamIndicator requests the server to send the *stream status* periodically upon which the server (SWC EMOSStreamingData) transmits *stream status request* and receives *stream status* via Ethernet stack. On receiving the *stream status*, it is sent to StreamActuator which further accesses I/O (LEDs) via ECUAbsSwComp as summarized in Table 5.4.

Table 5.4: Actuator component accessing I/O

Application Layer SWC	BSW Layer SWC	RTE Event	Interface	Data Elements
StreamActuator	ECUAbsSwComp	operationInvoked Event/serverCall-Point	C/S	LEDStatus

SWCs that have following ports have corresponding RTE Events to trigger runnables:

- Provide port - dataWriteAccess
- Receive port - dataReadAccess
- Client port - operationInvokedEvent
- Server port - serverCallPoint
- SWCs that accesses the underlying COM module via RTE gets triggered periodically using the RTE Event called *timing event.*

Basic software design

Basic Software was further designed and implemented for the above application as shown in Figure 5.5.

1. Application layer
   Application layer includes all the SWCs mentioned above encapsulated within a Composition SWC. SWCs were coded as per the above design using textual language ARText in Artic Studio tool. Further, ECU Extract was generated which was used to configure the underlying RTE and BSW layers.
2. RTE layer
   RTE layer was configured in RTE Builder tool of Arctic Studio by mapping runnables to OS tasks and instantiating the SWCs.
3. BSW layer
   BSW layer modules were further implemented and configured using Arctic Studio's BSW Builder tool. Although, only CAN part of the application was implemented initially. Since Arduino's CAN shield was an external device, an additional SPI driver was needed for accessing Arduino Mega 2560 CAN controller. As a result, external CAN Driver and SPI driver were manually implemented for Arduino hardware. Modules like OS, MCU, PORT and CANIf were partially implemented and configured. Further COM, Pdu Router, EcuM and DET were configured for Arduino Mega 2560.

This phase of implementation led to many challenges and roadblocks which are further described in the next section.

5.1.4 Challenges faced with the initial design and implementation

The challenges that were faced in this thesis are as described below (**RQ4.2**).

1. Modification of AUTOSAR OS
   Modifying the given AUTOSAR OS in Arctic Studio was not feasible because Arduino is a low-powered MCU which just runs a single program at a time and does not have an OS. Hence, a wrapper was written to interface FreeRTOS for Arduino and the AUTOSAR OS. A wrapper function is a layer of code that translates existing interface into a compatible

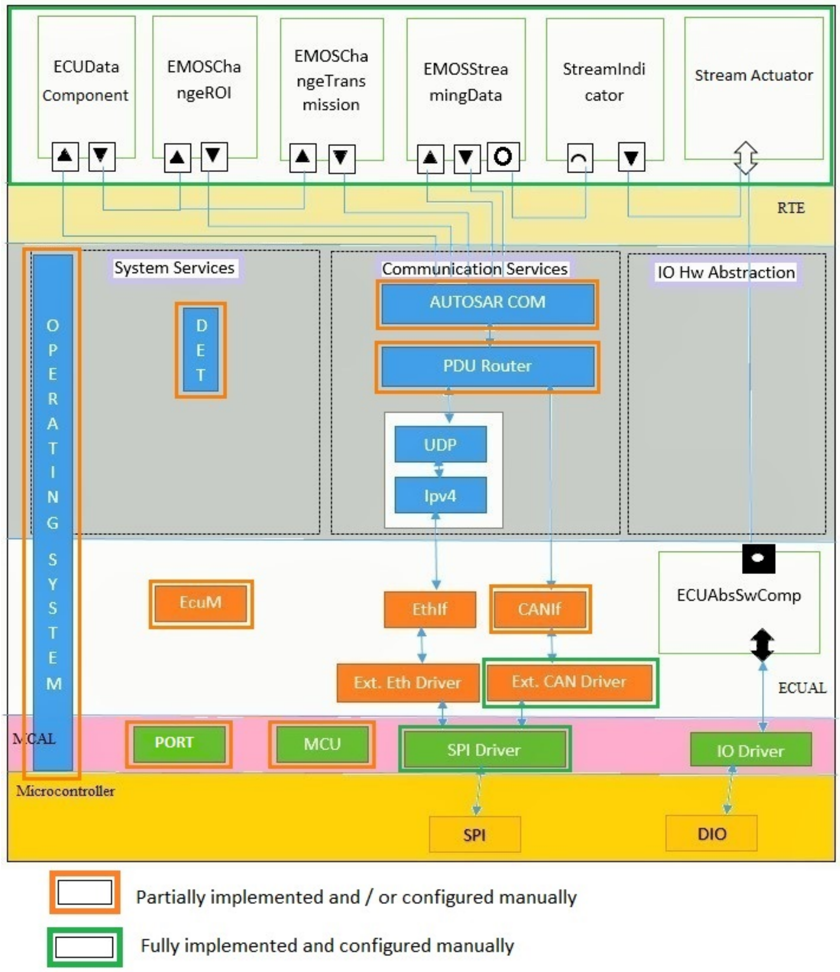


Figure 5.5: Initial design of the demonstrator

interface, in this case, existing AUTOSAR OS to a Arduino compatible FreeRTOS. Once the OS and other kernel source files were in place, it was time to compile the core AUTOSAR code.

2. Compiler for Arduino hardware *avr-gcc* not supported in Arctic Studio
   A compiler make file for avr-gcc was coded manually and integrated within the tool. This required multiple changes in other makefiles, which was cumbersome.

3. Lack of interoperability and integration with manually written code
   Since code generators were encrypted, newly created and partially modified modules and drivers could not be included within the generators to generate code automatically. As a result, all configuration files were statically modified.

4. Insufficient memory on Arduino hardware
   Though this was a concern while choosing the hardware, the initial strategy was to write our own code and to keep it simple in order to reduce the memory footprint. But as we progressed, tool offered less support in integrating code for Arduino and also resulted in memory mapping errors.

All the challenges mentioned above led us to conclude that, the standard AUTOSAR operating system provided by commercial tool vendors are not suitable for low-level micro-controllers unless specifically mentioned otherwise (for example MICROSAR from Vector GmBh). Hence, it was decided to change the hardware to yet another low-cost micro-controller from Texas Instruments and reduce the scope of application to just implement CAN communication and access I/O drivers.

5.2 Updated design of the demonstrator

Figure 5.6 shows updated conceptual design for the demonstrator.

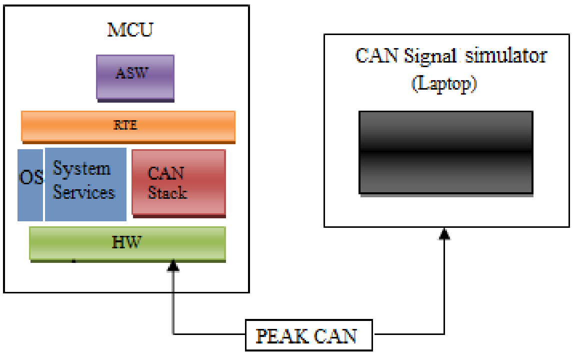


Figure 5.6: Final design of the demonstrator

5.2.1 System hardware

While PEAK CAN adapter and Lenovo laptop was used as previously decided, the following new hardware was procured as per the new design.

1. TMS570LC4357
   Hercules launchpad from Texas Instruments, TMS570LC4357 micro-controller, has 32-bit ARM Cortex R-5 dual processors which operates in lockstep clocking at 300MHz frequency. It has 128KB of data flash memory and 512KB of data RAM. It also has 4 on-chip DCAN controllers. Other communication channels include Ethernet, FlexRay, I2C, MibSPI and UART (SCI/LIN).

5.2.2 System software

The new application demonstrates an implementation of CAN communication and accessing the underlying I/O device. Figure 5.6 shows the new application design.

Functional Requirements

Among the previous set of Functional Requirements mentioned in Section 5.1.3, only CAN communication part of the implementation is being demonstrated in the updated design. As a result, the updated design has following functional requirements.

1. TMS570 hardware, should provide CAN communication interface based on the AUTOSAR architecture.

2. TMS570 should receive CAN messages to emulate steering angle and transmission status as inputs to the system. This also satisfies User Story 1 and User Story 2 (Section 5.1.3) partly.

3. TMS570 should access an on-board LED and blink periodically or stop blinking in order to emulate the stream signal.

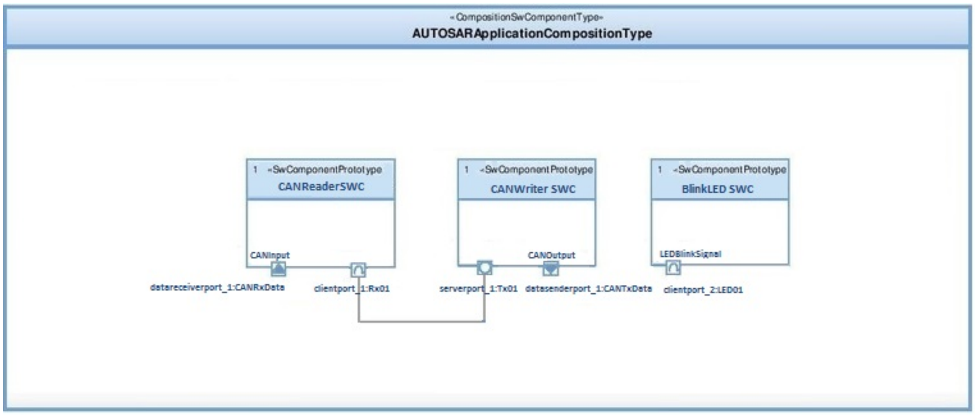


Figure 5.7: Final design of the demonstrator

Figures 5.7 and 5.8 shows the updated application and BSW design. CANReader SWC receives the CAN message from underlying COM module and writes it to CANWriter SWC. CANWriter SWC then transmits the received message to the hardware. The blinkLED SWC blinks LED light at a given time delay. Tables 5.5, 5.6 and 5.7 summarizes connections required to establish CAN communication and access I/O driver. All modules within the BSW layer were manually configured using Arctic Studio's BSW builder. Chapter 6 explains the implementation of this design in detail.

Table 5.5: Application design

Sender/Client SWCs	**Receiver/ Server SWCs**	**RTE Events**	**Interface**	**Data Elements**
CANReaderSWC	CANWriterSWC	operationInvoked Event/serverCall-Point	C/S	Rx01 / Tx01

Table 5.6: Communication matrix

I-Pdu	**SWCs**	**RTE Events**	**Interface**	**Data Elements**
Rx CAN signal	CANReaderSWC	dataWriteAccess/dataReadAccess	S/R	CANInput
Tx CAN Signal	CANWriterSWC	dataWriteAccess/dataReadAccess	S/R	CANOutput

Table 5.7: Actuator component accessing I/O

Application Layer SWC	**BSW Layer SWC**	**RTE Event**	**Interface**	**Data Elements**
BlinkLEDSWC	ECUAbsSwComp	operationInvoked Event/serverCall-Point	C/S	LEDBlinkSignal

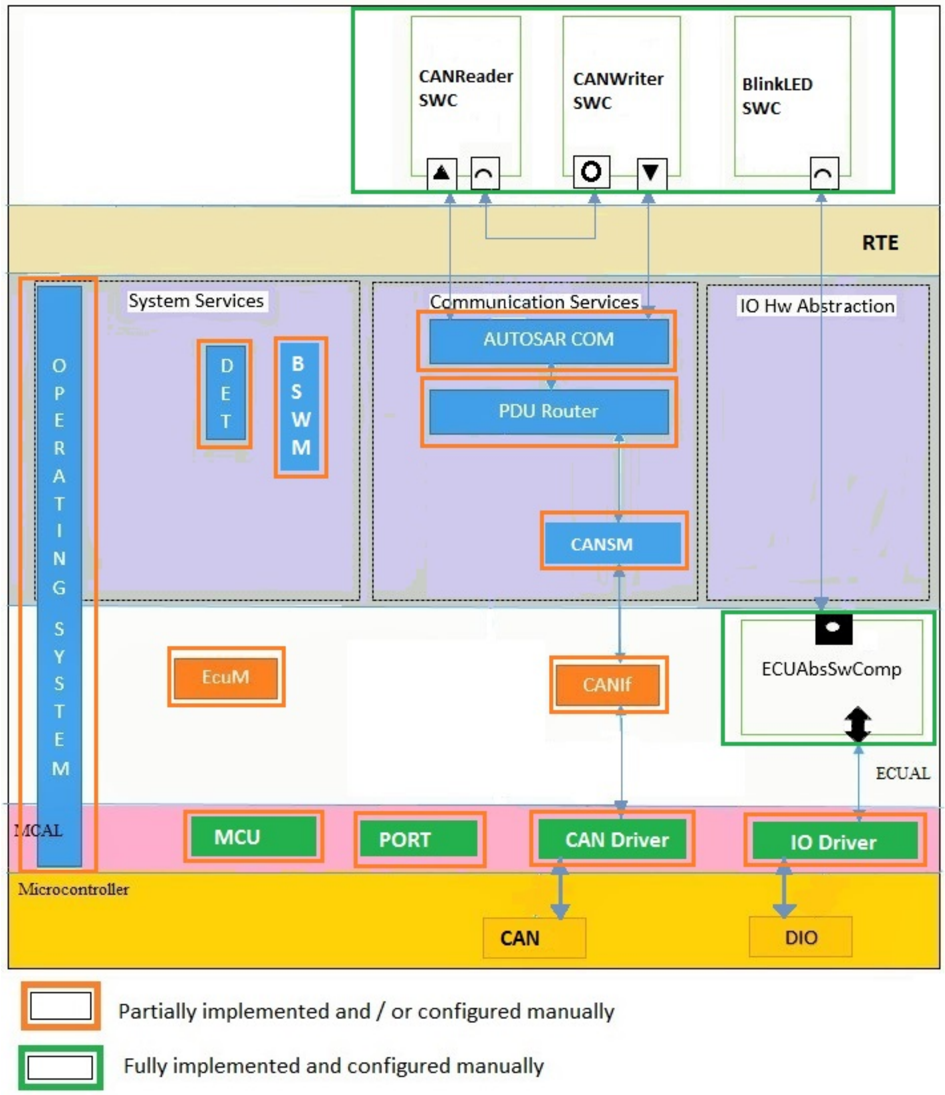


Figure 5.8: Final design of the demonstrator

5.3 Other tools used

Other tools used along with ArcCore's Arctic Studio include Code Composer Studio (CCS), AVRDude and IBM Rational Rhapsody. CCS tool from TI is used to flash the executable on TMS570LC4357 board. It is also used to debug the executable since this feature is not available within Arctic Studio. AVRDude programmer tool was used to flash the executable for Arduino Mega 2560 board. Application SWCs were designed using IBM Rational Rhapsody tool for AUTOSAR modeling.

Chapter 6

Implementation

6.1 Overview

This chapter focuses on realizing the demonstrator using Arctic Studio tool-chain. As discussed in the previous chapter, the board chosen initially was an Arduino Mega 2560. But due to many roadblocks and challenges (Section 5.1.4), it was decided that implementing AUTOSAR was not feasible on Arduino. Moreover the tool selected posed a major bottleneck for implementing AUTOSAR on an Arduino platform. With this we concluded that, using tools like Arctic Studio, it is not feasible to implement AUTOSAR on a low-powered micro-controller device (more on this in Chapter 8). Hence, a backup board was used to realize the demonstrator at a later stage.
The following chapter presents the implementation of an AUTOSAR compliant CAN communication stack on the Texas Instruments Hercules Launchpad (TMS570LC4357). The AUTOSAR tool from ArcCore, Arctic Studio is used to implement application, BSW layers and RTE layers and to generate an executable. Section 6.2 describes the implementation of application layer, section 6.3 presents the configuration of BSW modules using the BSW Editor tool and section 6.4 provides the configuration and generation of RTE layer using the RTE Editor tool. After the generation of all the configuration files (.c, .h), they are compiled together in order to generate the executable which is then flashed onto the hardware. Section 6.5 and 6.6 discusses the same in detail.

6.2 Application software development

To create the application layer, at first, SWC Description had to be defined. In order to facilitate the integration of software components, AUTOSAR provides a standard description format called the SWC Description (i.e. modeling the software components). An AUTOSAR Software Component Description (SWCD), describes the external structure and internal behavior of a software component. It gives an overall picture of the interfaces, ports, data elements, runnables, communication signals, software components definition and its internal behavior. A textual modeling language called ARText is provided in Arctic Studio to model the SWCs.
The creation of software components can be mainly divided into 3 phases. They are data and interface modeling phase, component modeling phase and writing the actual C code for implementing the SWC. These are described in the following sections.

6.2.1 Defining interfaces and data elements

In this phase, the interfaces which are being adopted in the application had to be defined along with the data elements and their types i.e. defining impl and app, as shown in Listing 6.1. Keywords *interface* is used. The two interfaces created in this application are i.e CanDataSRInterface which adopts sender receiver interface type and CanDataCSInterface that adopts client server interface type. Keywords *senderReceiver* and *clientServer* are used respectively, for this purpose.

```
// The impl, app and SR interface for communicating CANReader data
int impl CanDataImpl extends uint32
int app CanData

//Sender-Receiver interface
interface senderReceiver CanDataSRInterface {
  data CanData receive_message
  data CanData transmit_message
}

//Client-Server interface
interface clientServer CanDataCSInterface {
  operation Send {
    in CanData data1
  }
}

// Data Mappings for impl:s, app:s defined above
dataTypeMappingSet TypeMappings{
  map CanDataImpl CanData
}
```

Listing 6.1: Interfaces and data mapping SWCD code

Finally data and implementation mapping is done and is specified using *dataTypeMappingSet* keyword. Through this mapping, it is possible for ports to identify the intended data for a particular SWC and disregard other data.

6.2.2 Software Component Description (SWCD)

There are three SWCs implemented in this application (see Figure 5.5) and the software component descriptions are as specified below.

CANDataReader Component

Creating a SWCD has three important aspects i.e. creating the outer boundary for a SWC which includes ports and interfaces, defining an internal behavior for a SWC and creating an implementation for the internal behavior. An application component itself is created using the *component* keyword followed by name of the application as shown in the Listing 6.2. Further ports are created. Every SWC in AUTOSAR communicates via ports and all ports communicate over an interface (Section 6.2.1). CANDataReader component has a receiver port (RPort) through which it accepts data from the COM media (CanRxData) and hence requires the sender-receiver interface and a client port requests data from the server (CanClient) and hence adopts client server interface.

```
// Define the software components and Ports that use the defined interfaces
component application CanDataReaderComponent {
  ports {
    receiver CanRxData requires CanDataSRInterface
    client CanClient requires CanDataCSInterface
  }
}
internalBehavior CanDataReaderBehaviour for CanDataReaderComponent {

  dataTypeMappings {
    TypeMappings
  }

  runnable CanDataReaderComponentMain [0.0] {
    symbol "CanDataReaderComponentMain"

    dataReadAccess CanRxData.receive_message
    serverCallPoint synchronous CanClient.*

    dataReceivedEvent CanRxData.receive_message
  }
}
implementation CanDataReaderImpl for CanDataReaderBehaviour {
  language c
  codeDescriptor "src"
}
```

Listing 6.2: CANDataReaderComponent SWC

Next step is to define the functionality of SWCs and this is done by creating an internal behavior which includes runnables for the CANDataReaderComponent. Internal behavior is created using *internalBehavior* keyword for CANDataReaderComponent and runnable is created using the keyword *runnable*. All the runnables in this application are instantiated just once, but multiple instantiation of a single runnable is also possible.
Since a receiver port is implemented within this SWC, the runnable requires data read access and the syntax for this is *dataReadAccess*. Hence, CanRxData of the receiver port is mapped to the receive_message data that the SWC is expected to receive. The client port makes a synchronous call to the server port using the command *serverCallPoint*. If a runnble has no data access mentioned explicitly in order to communicate over a certain port, then an API for communication will not be generated for such a runnble by RTE. This runnable is triggered by data received event i.e. when a CAN message is received.
The third part of defining a SWC is to specify an implementation for its runnable which is coded in C language in a separate .c file. This file will be linked to the SWC during the generation of RTE.

CANDataWriter Component

CANDataWriter component writes data to the COM media and hence makes use of PPort CANTxData. It uses SR interface for this purpose. This SWC also acts a server for the CANReaderData component and uses server port CanServer via the CS interface as shown in Listing 6.3.

```
// Define the software components and Ports that use the defined interfaces
component application CanDataWriterComponent {
  ports {

    sender CanTxData provides CanDataSRInterface
    server CanServer provides CanDataCSInterface
  }
}
internalBehavior CanDataWriterBehaviour for CanDataWriterComponent {
  dataTypeMappings {
    TypeMappings
  }

  runnable CanDataWriterComponentMain [0.0] {
    symbol "CanDataWriterComponentMain"

    dataWriteAccess CanTxData.transmit_message

    operationInvokedEvent CanServer.Send
  }
}
implementation CanDataWriterImpl for CanDataWriterBehaviour {
  language c
  codeDescriptor "src"
}
```

Listing 6.3: CANDataReaderComponent SWC

The internal behavior includes a runnable that has data write access through which RTE provides mechanism that allows this component to write data to COM media and operation invoked event invokes the server when client makes a call. Implementation for the runnable is coded in C as a separate C file which gets linked during run-time.

BlinkLED component

BlinkLED component acts as an actuator component which is used to access the ECU's I/O. It includes a client port that establishes contact directly with the server port of an ECU abstraction software component. This component is triggered periodically by making use of timing event as shown in Listing 6.4.

```
// Define the software components and Ports that use the defined interfaces
component application BlinkLEDComponent {
  ports {
    client LEDDigitalLight requires DigitalServiceWrite
  }
}

internalBehavior BlinkLEDBehaviour for BlinkLEDComponent {

  dataTypeMappings {
    TypeMappings
  }

  runnable BlinkLEDComponentMain [0.0] {
    symbol ”BlinkLEDComponentMain”

    serverCallPoint synchronous LEDDigitalLight.*

    timingEvent 0.1 as blinkLEDMainEvent
  }
}

implementation BlinkLEDImpl for BlinkLEDBehaviour {
  language c
  codeDescriptor ”src”
}
```

Listing 6.4: BlinkLEDComponent SWC

6.3 ECU extract generation

Once all the SWCs are created, an extract (ARXML file) for a single ECU is generated and is used for configuring the ECU. This extract includes all the information related to the application components and service components required from the underlying BSW layer i.e. BSWM and ECUM.

Service components provide access to the SWCs to use functions provided by the BSW modules. For example, BSWM component allows an application to access PDU groups that needs to be sent, ECUM provides an access for application components to manage ECU modes (which inturn communicates with the corresponding service module in BSWM) and also handles the initialization of ECU. IoHwAb is an ECU Abstraction SWC that provides direct IO access for a SWC to the DIO driver and is responsible for imparting digital IO services to the application layer. Hence, this is also vaguely considered as a service component though its not and is part of the ECU Abstraction Layer.

6.4 Developing the application code

This section explains the actual C code implementation for the SWCs. During RTE generation using the RTE Editor tool, SWC implementations are created by RTE by creating a function called Rte_IRead_CanDataReaderComponentMain_CanRxData_receive_message(). This function returns a pointer which points to the received data on the RPort and is stored in rx_message. Further, Rte_Call_CanClient_Send() is used to invoke the server (CANDataWriterComponent) by passing the received message (rx_message) as a parameter, as shown in Listing 6.5.

```
// Receive CAN message
rx_message = Rte_IRead_CanDataReaderComponentMain_CanRxData_receive_message ();

// Call server
Rte_Call_CanClient_Send (rx_message);
```

Listing 6.5: C code implementation for CAN data reader component

Listing 6.6. shows the implementation for CANDataWriterComponent. RTE generates a function call Rte_IWrite_CanDataWriterComponentMain_CanTxData_transmit_message() where the received message from the client server interface is passed as a parameter to write to COM media via its PPort.

```
// Transmit the received CAN message
Rte_IWrite_CanDataWriterComponentMain_CanTxData_transmit_message (client_message);
```

Listing 6.6: C code implementation for CAN data writer component

In order to actuate the LED, it generates Rte_Call_LEDDigitalLight_Write() by passing the level as a parameter. Led_on is assigned with a value 1 which is then toggled to make it blink at each second as shown in Listing 6.7.

```
void BlinkLEDComponentMain (void) {
  Rte_Call_LEDDigitalLight_Write (led_on);

  led_on = !led_on;
  }
```

Listing 6.7: C code for implementing BlinkLED software component

6.5 ECU configuration

Configuring an ECU is the most complex process while developing an AUTOSAR application. Arctic Studio provides BSW Editor tool for this process. As shown in Figure 6.1, this tool takes ECUExtract.arxml (the ARXML file generated in the previous step as part of an ECU extract). The generator creates configuration files (*_Cfg.h, *_Cfg.c, *.mk) by making use of static source files from Arctic Core (the BSW core project of Arctic Studio).

With the help of BSW Editor tool, required BSW modules for implementing the above application, had to be configured. As shown in Figure 6.2 only required BSW modules are added for configuration.

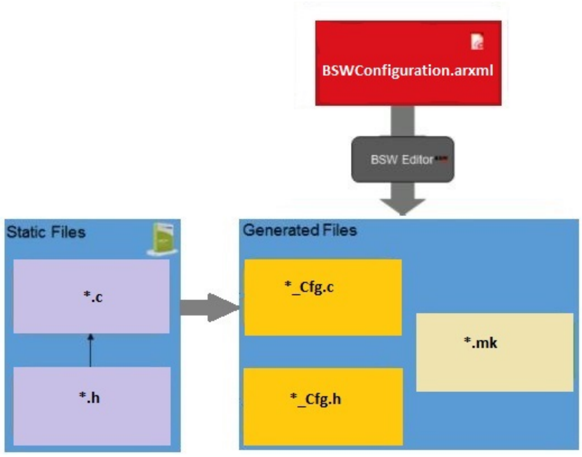
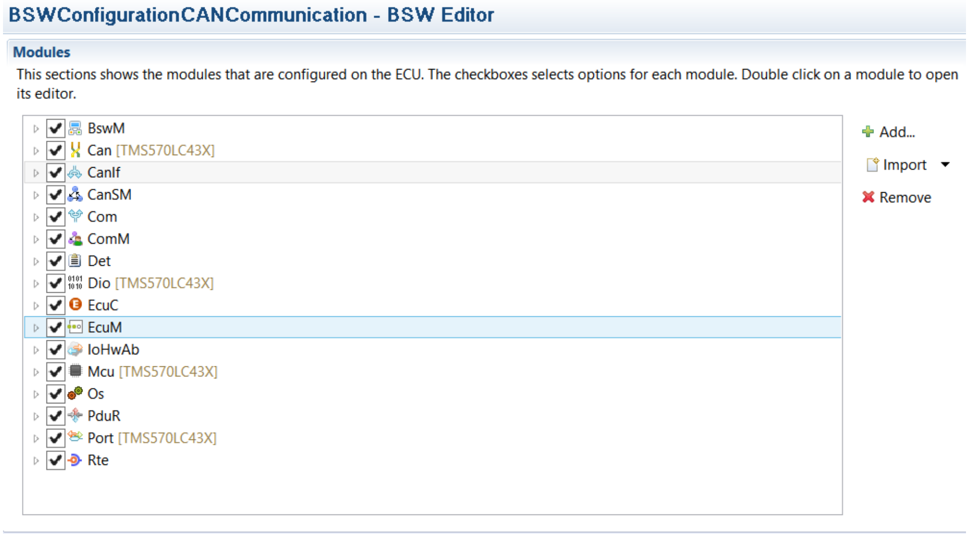


Figure 6.1: Generation of configuration files using BSW Editor tool



BSWConfigurationCANCommunication - BSW Editor

Modules

This sections shows the modules that are configured on the ECU. The checkboxes selects options for each module. Double click on a module to open its editor.

- ☑ BswM
- ☑ Can [TMS570LC43X]
- ☑ CanIf
- ☑ CanSM
- ☑ Com
- ☑ ComM
- ☑ Det
- ☑ Dio [TMS570LC43X]
- ☑ EcuC
- ☑ EcuM
- ☑ IoHwAb
- ☑ Mcu [TMS570LC43X]
- ☑ Os
- ☑ PduR
- ☑ Port [TMS570LC43X]
- ☑ Rte

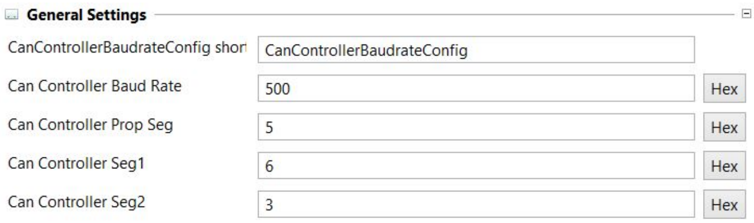
Figure 6.2: BSW Editor tool interface in Arctic Studio

6.5.1 CAN driver

A layered architecture style builds from lower layers to upper layers. Moreover, upper layer modules extends the functionality of lower layer modules. Hence, first the lower layer modules must be configured followed by the modules in the layer above, i.e. CAN driver is configured followed by CANIF, CANSM, PduR, COMM and COM module. In order to configure the CAN driver, CAN controller baud rate and CAN hardware receive and transmit objects must be configured as shown in the Figure 6.3 and 6.4.

6.5.2 CAN interface

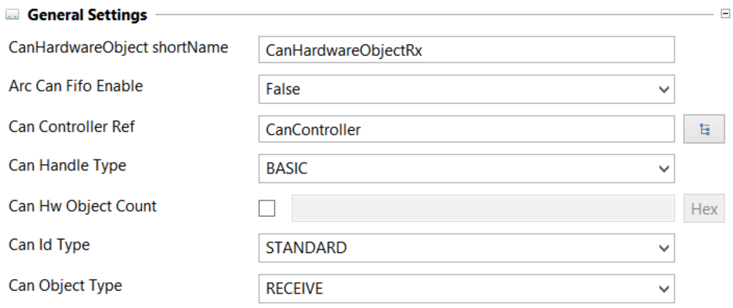
CANIF module requires configuring CANIF Pdus i.e CANRxPduCfg and CANTxPduCfg as shown in Figure 6.5 (only configurations for Rx is shown in figures). Here it also establishes a reference to the CAN hardware object created in CAN driver i.e. CAN hardware object receive handle (CanIfHrhCfg, CanIfHthCfg) is referenced to the CAN hardware receive object as shown in Figure 6.6.



General Settings

CanControllerBaudrateConfig shor	CanControllerBaudrateConfig	
Can Controller Baud Rate	500	Hex
Can Controller Prop Seg	5	Hex
Can Controller Seg1	6	Hex
Can Controller Seg2	3	Hex

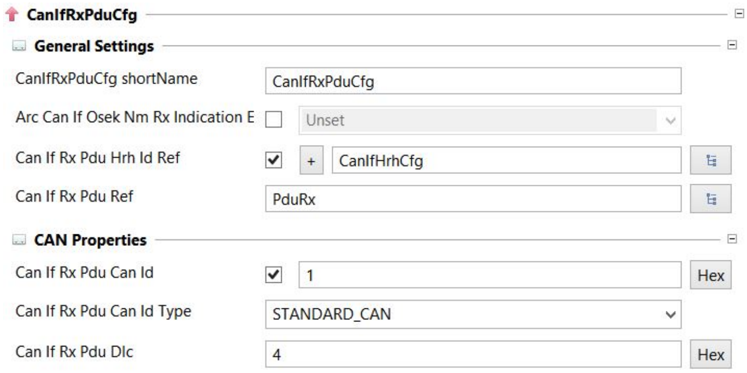
Figure 6.3: Configuring CAN Driver in Arctic Studio



General Settings

CanHardwareObject shortName	CanHardwareObjectRx	
Arc Can Fifo Enable	False	
Can Controller Ref	CanController	
Can Handle Type	BASIC	
Can Hw Object Count	□	Hex
Can Id Type	STANDARD	
Can Object Type	RECEIVE	

Figure 6.4: Configuring CAN Driver in Arctic Studio



CanIfRxPduCfg

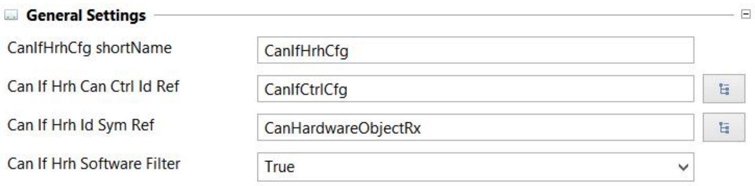
General Settings

CanIfRxPduCfg shortName	CanIfRxPduCfg	
Arc Can If Osek Nm Rx Indication E	□ Unset	
Can If Rx Pdu Hrh Id Ref	☑ + CanIfHrhCfg	
Can If Rx Pdu Ref	PduRx	

CAN Properties

Can If Rx Pdu Can Id	☑ 1	Hex
Can If Rx Pdu Can Id Type	STANDARD_CAN	
Can If Rx Pdu Dlc	4	Hex

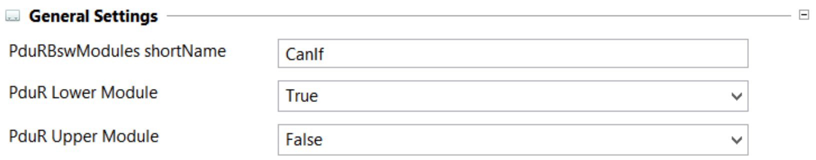
Figure 6.5: Configuring CANIF module - CanIfRxPdu being assigned to CanIfHrhCfg



General Settings

CanIfHrhCfg shortName	CanIfHrhCfg
Can If Hrh Can Ctrl Id Ref	CanIfCtrlCfg
Can If Hrh Id Sym Ref	CanHardwareObjectRx
Can If Hrh Software Filter	True

Figure 6.6: CanIfHrhCfg being referenced to CanHardwareObjectRx



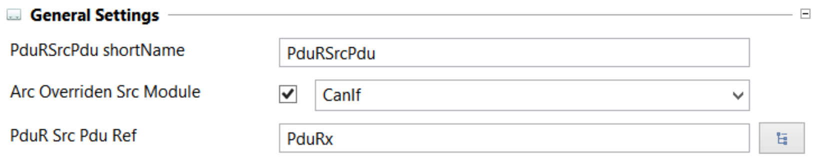
General Settings

PduRBswModules shortName	CanIf
PduR Lower Module	True
PduR Upper Module	False

Figure 6.7: PDU routing table

6.5.3 PduR

For configuring the PduR module, routing table must be configured as shown in Figure 6.7 and 6.8. First, BSW modules used by PduR must be mentioned. In this case PduR communicates with COM module and CANIF module. Figure 6.7 shows the communication hierarchy i.e. CANIF being a lower level module to PduR. Similarly COM is configured as upper module. Further, CANIF is designated to be a SrcPdu (as shown in Figure 6.8) and COM is assigned to be a destination Pdu, when the message is received from lower layers to upper layers. Likewise, vice versa i.e. COM is assigned as source Pdu and CANIF is assigned as a destination Pdu when the message is being transmitted.



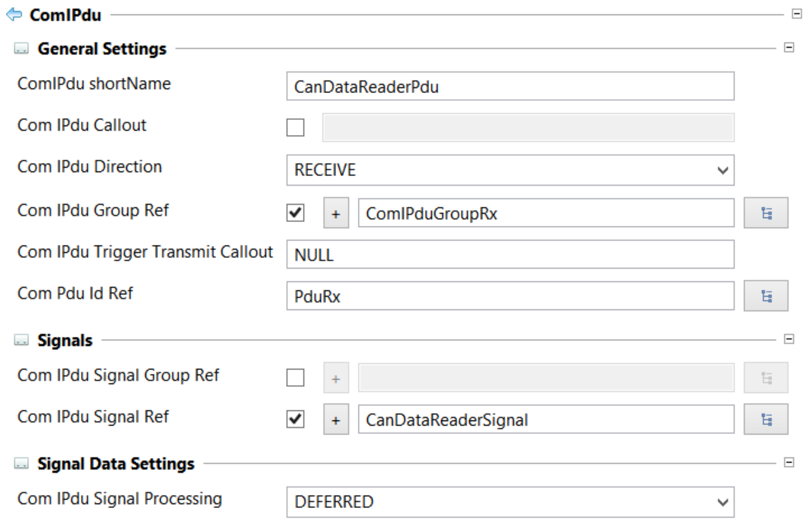
General Settings

PduRSrcPdu shortName		PduRSrcPdu
Arc Overriden Src Module	☑	CanIf
PduR Src Pdu Ref		PduRx

Figure 6.8: CANIF assigned to be source PDU while CAN message is received

6.5.4 COM

There were three main configurations that were made within the COM module for this application. They are, configuring instruction Pdu (I-Pdu), configuring I-Pdu groups and configuring the COM signals. An I-Pdu contains the data message that has been either received from one of the communication modules in the communication stack, or a data message that is to be sent to one of said modules. An I-PDU also belongs to an I-PDU group i.e ComIpduGroupRx as shown in Figure 6.9. The data of an I-PDU is divided into signals, depending on bit position in the I-PDU. For example if you have an I-PDU with 2 bytes of data length and desire to place the first byte to one signal and second byte to another signal, you set the bit position of each signal accordingly. The data length used in this case is 4 bytes and hence can be sent over a single COM signal (max 4 bytes). If there are more than 4 bytes that needs to be sent then signal groups can be created where the IPdus are grouped into signal groups.



ComIPdu

General Settings

ComIPdu shortName		CanDataReaderPdu
Com IPdu Callout	□	
Com IPdu Direction		RECEIVE
Com IPdu Group Ref	☑ +	ComIPduGroupRx
Com IPdu Trigger Transmit Callout		NULL
Com Pdu Id Ref		PduRx

Signals

Com IPdu Signal Group Ref	□ +	
Com IPdu Signal Ref	☑ +	CanDataReaderSignal

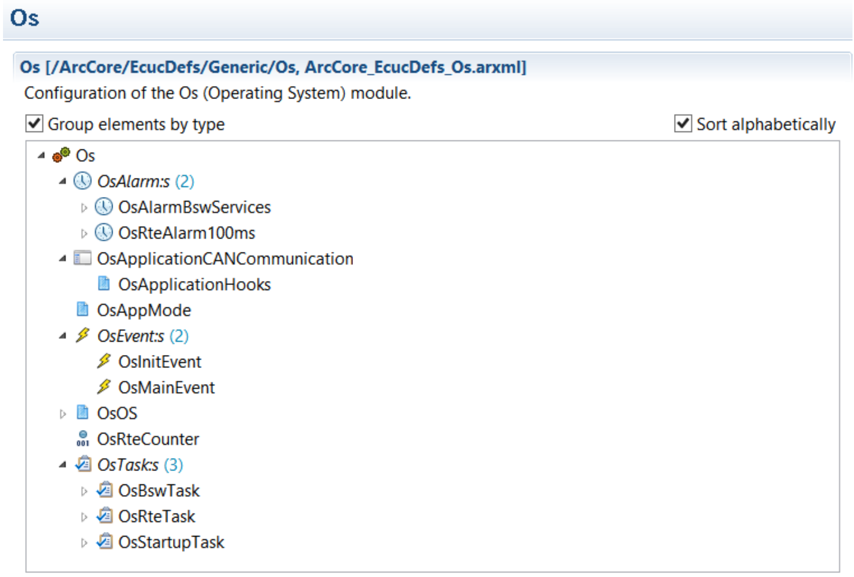
Signal Data Settings

Com IPdu Signal Processing	DEFERRED

Figure 6.9: COM module configuration

6.5.5 AUTOSAR OS

Further, service layer modules like OS, BswM and EcuM that imparts system services to the application were to be configured. The BswM and EcuM modules provide services to the application components to interact with the BSW modules, which is further explained in the sub sections below.
In order to configure AUTOSAR OS, five main entities that are configured in this case, they are, Alarms, Events, Counter, Tasks and OsApplication as shown in Figure 6.9. A counter is an entity that keeps count in the OS module which can be set to be handled either in software, hardware or according to OS ticks. In this case Os tick frequency is configured to be 100 within OsOs parameter. An event is an entity which is referred by a task. Event facilitates the triggering of runnable by the RTE as explained in section 6.2. For this purpose RTE makes use of OS Alarms which is used to trigger the events at respective time intervals. A task is an entity like events also triggered by alarms. In this case three tasks are created OsRteTask, OsBswTask and OsStartupTask. The configuration of these tasks are explained in Section 6.6 under RTE configuration. An OsApplication entity simply acts as a container for counters, tasks and events for a particular application.



Os

Os [/ArcCore/EcucDefs/Generic/Os, ArcCore_EcucDefs_Os.arxml]

Configuration of the Os (Operating System) module.

☑ Group elements by type ☑ Sort alphabetically

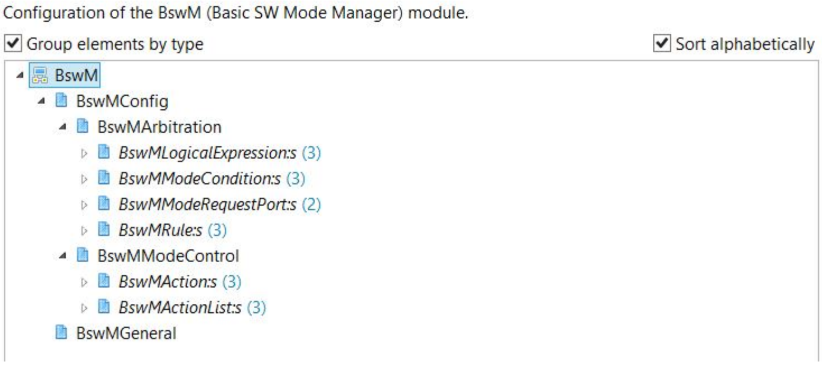
```
Os
  OsAlarm:s (2)
    OsAlarmBswServices
    OsRteAlarm100ms
  OsApplicationCANCommunication
    OsApplicationHooks
  OsAppMode
  OsEvent:s (2)
    OsInitEvent
    OsMainEvent
  OsOS
  OsRteCounter
  OsTask:s (3)
    OsBswTask
    OsRteTask
    OsStartupTask
```

Figure 6.10: Configuration of AUTOSAR OS

6.5.6 BswM

BswM module is mainly used to access services provided by the BSW modules through application layer. There are two main parts while configuring BswM module, namely, BswMArbitration and BswMModeControl as shown in Figure 6.11. BswMArbitration is a container that contains BswMLogicalExpressions, BswMModeConditions, BswMModeRequestPorts and BswMRules. BswM provides three main rules within BswMRules container viz., DisablePduGroup, EnablePduGroup and StartCommunicationRule. Using these logical expressions viz., ComMFullCommunicationalExpression, ComMNoCommunicationalExpression, StartCommunicationExpression are evaluated. Logical expressions are created by chaining together different mode conditions using common logical operators (AND, NAND, OR, XOR). BswMModeControl container has BswMActions and BswMActionLists. Actions are grouped in action lists. Action specifies how the I-PDUs are treated in a specific mode (acts like a group switch).
There are three actions created within the BswMAction container. These are IPduAllDisabled

(both Rx IPdus and Tx IPdus are disabled), IPduAllEnabled (both Rx IPdus and Tx IPdus are enabled), RequestFullCom (through this parameter a full communication access to the respective communication channel can be requested). A BswMActionList container is a list of one or more BswMActionListItems with an addition field specifying how the action list is executed. A BswmActionListItem allows the configurator to specify the exact position of the items in the list and even offer the possibility of referencing other ActionLists and even Rules.



Configuration of the BswM (Basic SW Mode Manager) module.

☑ Group elements by type ☑ Sort alphabetically

- BswM
  - BswMConfig
    - BswMArbitration
      - *BswMLogicalExpression:s* (3)
      - *BswMModeCondition:s* (3)
      - *BswMModeRequestPort:s* (2)
      - *BswMRule:s* (3)
    - BswMModeControl
      - *BswMAction:s* (3)
      - *BswMActionList:s* (3)
  - BswMGeneral

Figure 6.11: Configuration of BswM module

6.5.7 EcuM

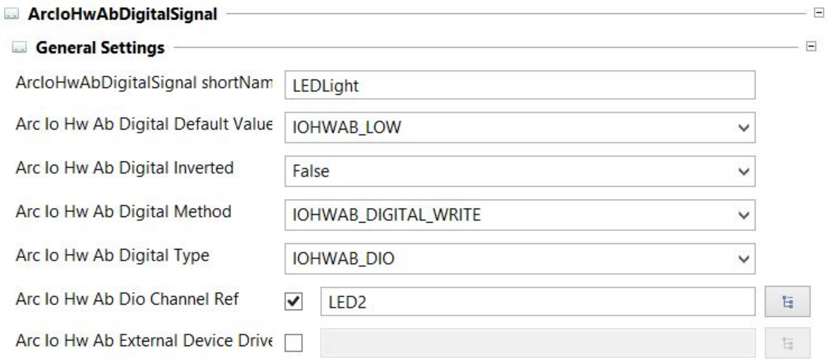
The main function of EcuM module is to initialize the ECU and manage the ECU states. EcuM has fixed configurations viz., STARTUP, RUN, SLEEP, WAKEUP, SHUTDOWN which means that the states of EcuM are fixed and are predefined by AUTOSAR and hence cannot be altered. AUTOSAR has introduced EcuM flex, where an user can change the states, from AUTOSAR v4.x. In this case, EcuM flex and BswM are closely coupled and work together.

6.5.8 IoHwAb

Next, I/O modules such as IoHwAb, PORT and DIO modules had to be configured. The IO Hardware Abstraction module abstracts the signal path of the ECU hardware (Layout, Microcontroller Pins, Microcontroller external devices like IO ASIC). It provides a signal based interface to the upper software layer. It performs static abstraction and inversion (if needed) of values according to their physical representation at the inputs/outputs of the ECU hardware. In this case, a channel reference is established to LED2 so that the SWC in application layer can communicate with the DIO module directly. Since, LED is considered as an actuator, it is set to digital write, as shown in Figure 6.12.

6.5.9 PORT

The PORT module configures hardware pins. The DIO module can then perform read and write operations based on these initial configurations. The configuration involve initializing the values on the pins, setting a direction and mode for each pins as shown in Figure 6.13. On power-up the pins in a micro-controller are set to a default value. The port driver initializes all the pins either as low or high as per the system requirements. In this case, the port pin is pulled up. A pull up or a pull down resistor is used so that there are no floating connections. Pins can be either input or output. Sometimes system requirement will make it necessary to change the direction of pins. Other modules can read to pins configured as input and write to pins configured as output and can request to change the pin direction only if they are configured for that in the Port module. Based on the functionality of the pins, their mode is configured. In this case, the mode is configured to DIO to support LED2 pin.

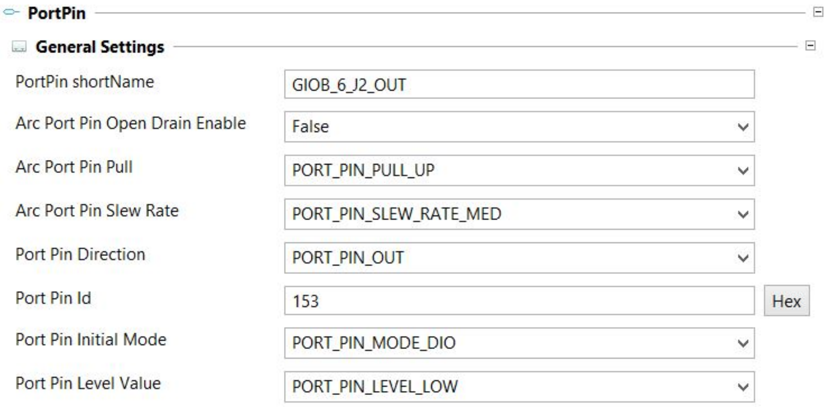


ArcIoHwAbDigitalSignal

General Settings

ArcIoHwAbDigitalSignal shortNam	LEDLight
Arc Io Hw Ab Digital Default Value	IOHWAB_LOW
Arc Io Hw Ab Digital Inverted	False
Arc Io Hw Ab Digital Method	IOHWAB_DIGITAL_WRITE
Arc Io Hw Ab Digital Type	IOHWAB_DIO
Arc Io Hw Ab Dio Channel Ref	☑ LED2
Arc Io Hw Ab External Device Drive	□

Figure 6.12: Configuration of IoHwAb module



PortPin

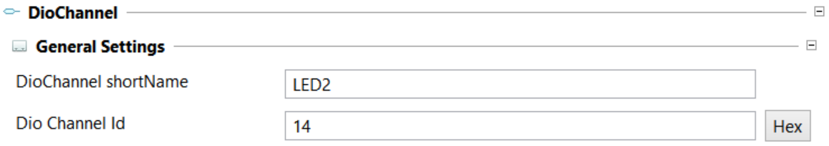
General Settings

PortPin shortName	GIOB_6_J2_OUT
Arc Port Pin Open Drain Enable	False
Arc Port Pin Pull	PORT_PIN_PULL_UP
Arc Port Pin Slew Rate	PORT_PIN_SLEW_RATE_MED
Port Pin Direction	PORT_PIN_OUT
Port Pin Id	153
Port Pin Initial Mode	PORT_PIN_MODE_DIO
Port Pin Level Value	PORT_PIN_LEVEL_LOW

Figure 6.13: Configuration of PORT module

6.5.10 DIO

The DIO driver works on pins and ports which are configured by the PORT module. A physical input or output pin on an MCU device is in PORT module called port pin and in DIO module a port pin is represented by a DIO channel as shown in Figure 6.14. Port pins need to be initialized by PORT module before being used in DIO module. They will also have the same ID in both PORT and DIO modules. IoHwAb references DIO channel by name (Figure 6.12).



DioChannel

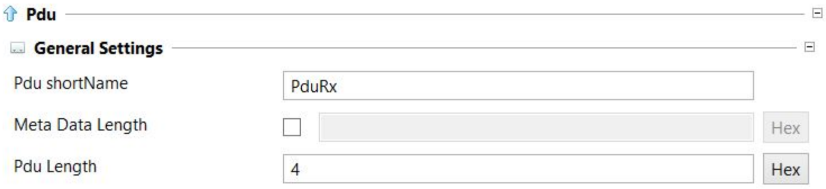
General Settings

DioChannel shortName	LED2
Dio Channel Id	14

Figure 6.14: Configuration of DIO module

6.5.11 EcuC

EcuC module is used to create Pdu objects which is required by the communication stack modules. As shown in Figure 6.15 and 6.16, each CAN message requires a separate PDU and also needs to be created in both the directions (receive, transmit) i.e. PduRx and PduTx. The data length is configured to be 4 bytes.



Pdu

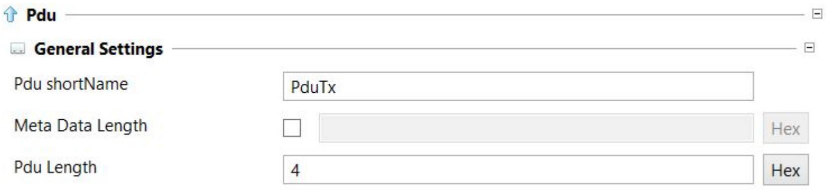
General Settings

Pdu shortName: PduRx

Meta Data Length: □ Hex

Pdu Length: 4 Hex

Figure 6.15: Configuration of EcuC module



Pdu

General Settings

Pdu shortName: PduTx

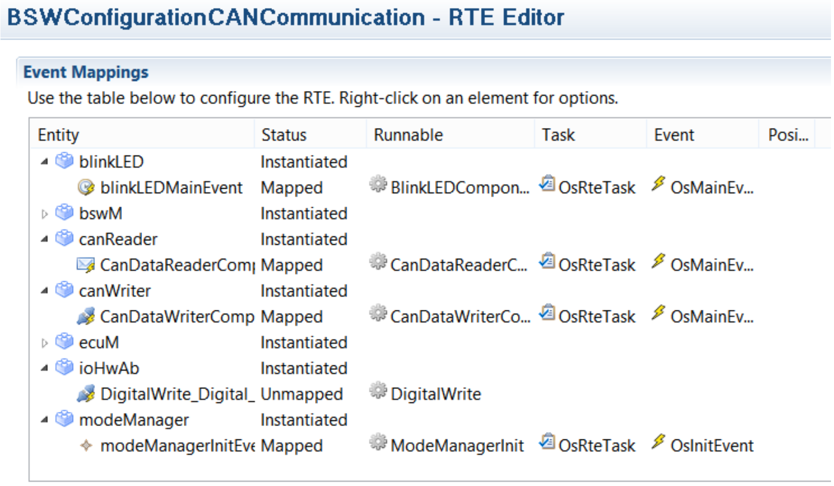
Meta Data Length: □ Hex

Pdu Length: 4 Hex

Figure 6.16: Configuration of EcuC module

6.6 RTE configuration

RTE layer facilitates communication between application SWCs and between SWCs and BSW modules. The first step in RTE configuration is to instantiate all RTE entities, as shown in Figure 6.17. Next, all runnables are be mapped to OsRteTask which can be triggered by RTE Events and hence all the SWCs are assigned to OsMainEvent which triggers the runnables based on the triggered events mentioned in SWC internal behavior. Further, all the service tasks (ECUM, BSWM, IoHwAb) are mapped to OsBswTask internally. Hence, they are left unmapped while configuring RTE. Finally, RTE is generated and validated.



BSWConfigurationCANCommunication - RTE Editor

Event Mappings

Use the table below to configure the RTE. Right-click on an element for options.

Entity	Status	Runnable	Task	Event	Posi...
blinkLED	Instantiated				
blinkLEDMainEvent	Mapped	BlinkLEDCompon...	OsRteTask	OsMainEv...	
bswM	Instantiated				
canReader	Instantiated				
CanDataReaderCom|	Mapped	CanDataReaderC...	OsRteTask	OsMainEv...	
canWriter	Instantiated				
CanDataWriterComp	Mapped	CanDataWriterCo...	OsRteTask	OsMainEv...	
ecuM	Instantiated				
ioHwAb	Instantiated				
DigitalWrite_Digital_	Unmapped	DigitalWrite			
modeManager	Instantiated				
modeManagerInitEv	Mapped	ModeManagerInit	OsRteTask	OsInitEvent	

Figure 6.17: RTE configuration

6.7 Generation of executable

After configuring all the BSW modules and RTE, each module should be generated and validated. This is done in Arctic Studio by just clicking on generation and validation tab. Further, the target is made. After compiling all the configuration files, source files and post-build configuration files, an .elf binary file is generated.

```
Image size: (decimal)
  text:       59496 B      58.1 kB
  data:        1048 B       1.0 kB
  bss:        11512 B      11.2 kB
  ROM:        60544 B      59.1 kB
  RAM:        12560 B      12.3 kB
```

Listing 6.8: .elf file execution output

Listing 6.8 shows that RAM size and ROM size are 12.3kB and 59.1 kB respectively. This looked a bit surprising initially because in reality an AUTOSAR program consumes more space and is memory intensive. But it could be found that the tool optimizes the code and hence the above results.

6.8 Flashing on hardware

The executable generated in previous step (.elf binary file) was flashed on the hardware. Figure 6.17 shows CAN Hi and CAN Lo pins of Peak CAN adapter connected to DCAN2RX, DCAN2TX controllers of TMS570LC4357.

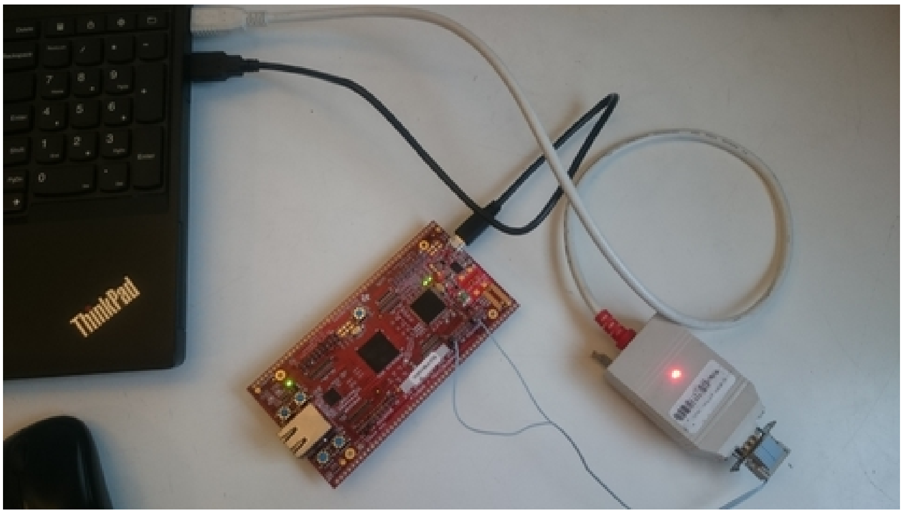


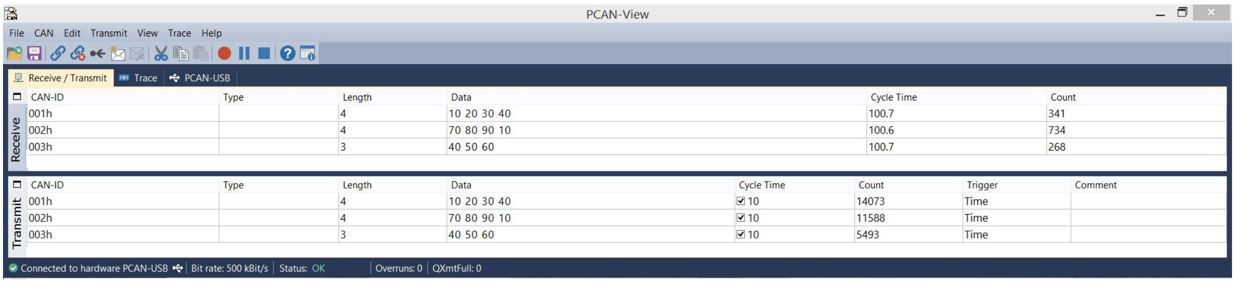
Figure 6.18: Flashing the binaries on TI Hercules TMS570LC4357 Launchpad

6.8.1 Results

The results demonstrate the implementation of CAN communication interface using the AUTOSAR architecture. Accordingly, user stories as specified in Section 5.1.3 have been implemented and is as shown in Figure 6.19 and 6.20.

1. CAN communication
   Figure 6.19 demonstrates the messages being sent and received over the PEAK CAN dongle using the underlying AUTOSAR architecture. According to User Stories 1 and 2, AUTOSAR ECU was required to receive CAN messages. This basic functionality has been demonstrated.

2. Activation of LED2
   Further, User Story 3 required the demonstrator to access an on-board LED which acts as an indicator. Accordingly, this requirement has been satisfied and is as shown in Figure 6.20.



CAN-ID	Type	Length	Data	Cycle Time	Count
001h		4	10 20 30 40	100.7	341
002h		4	70 80 90 10	100.6	734
003h		3	40 50 60	100.7	268

CAN-ID	Type	Length	Data	Cycle Time	Count	Trigger	Comment
001h		4	10 20 30 40	☑ 10	14073	Time	
002h		4	70 80 90 10	☑ 10	11588	Time	
003h		3	40 50 60	☑ 10	5493	Time	

Connected to hardware PCAN-USB | Bit rate: 500 kBit/s | Status: OK | Overruns: 0 | QXmtFull: 0

Figure 6.19: CAN Transmit and Receive messages

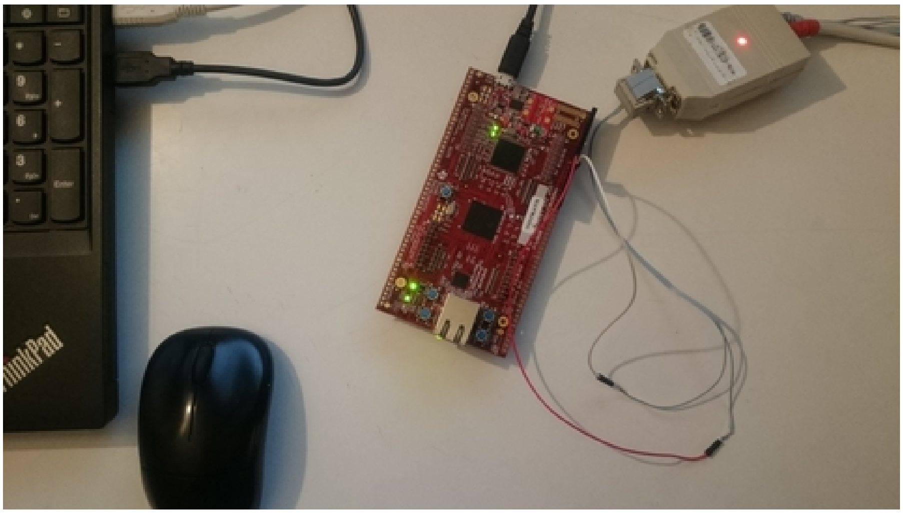


Figure 6.20: Activated LED light

Chapter 7

Evaluation

This chapter provides an evaluation of the ArcCore tool-chain used to implement the demonstrator in Chapter 6. Further, the efficiency of the AHP algorithm for this application is analyzed.

7.1 Evaluation of the ArcCore Tool-chain

In this section, the Arctic Studio tool-chain (which includes Application Development Tool, BSW Editor and RTE Editor) along with Arctic Core (which provides BSW modules and MCAL modules) are evaluated based on the criteria and sub-criteria used for selection of AUTOSAR tools in Chapter 4 (Figure 4.10). The experience gained in implementing the demonstrator in Chapter 6, led to the following main observations about this tool-chain from ArcCore.

1. Functionality

    - *System modeling and modeling analysis*
      Arctic Studio's application development tool, adopts ARText, which is a powerful textual language for modeling the SWCs. Although using ARText framework has many advantages like auto completion of code, syntax highlighting, integrated validation, version control, documentation etc., a graphical interface to modeling has its own benefits. In order to develop functions with many SWCs, a graphical programming tool helps in managing the complexity better than textual programming. Since the demonstrator for this case was fairly simple, modeling functionalities included within Arctic Studio was sufficient. However, it might not be a suitable option for systems with many SWCs as the tool does not provide a GUI of it's own (i.e. if one requires GUI, then they should also make use of Embedded Coder from Mathworks for instance).

      Arctic Studio also does on-the-fly transformation of the model being developed to an AUTOSAR model. This way it is easy to analyze if the model is in sync with the ARText file being developed.

    - *Network modeling and timing analysis*
      ARText language also supports network modeling and timing analysis. It provides interfaces for different types of connectors, ports and interfaces between SWCs. As mentioned above, if the system is complex and has many SWCs, then network connections become cumbersome to be managed using only textual language and provision for having a graphical programming environment would save a lot of time.

      Further, the ECU Extract file generated by Arctic Studio also includes communication information of the underlying BSW modules. This information is fully specified as a part of the system design and are represented as communication matrices in the system description ARXML files. *Arctic Studio's BSW Editor tool* then reads the system description file and automatically includes the data elements which needs to be configured while configuring ECU. This process is quite easy using this tool and saves a lot of time when compared to manual implementation as in ECU-centric approach.

- *System configuration and code generation*
  Application code generation is done automatically with the help of proprietary code generators, although, runnables have to be implemented manually.

  As mentioned above, configuring BSW modules is done using *BSW Editor tool.* If there are any errors or incompatible values entered during configuration phase, the tool provides a validator to validate all the configured modules. After configuring and validating all the required modules, the tool generates code automatically. One drawback to this approach is, it is not possible to integrate BSW modules that are manually created into this tool since Arctic Studio does not provide access to source code.

  Arctic Studio's RTE Editor tool provides an interface to instantiate and map the runnables to specific OS tasks. This mapping is fairly easy but the whole problem arises during code compilation since there is no way to debug the code until the end or for separate modules.

  ArcCore's Arctic Core includes code for specific hardware boards. Although, not all of the modules are fully developed. For example, not all hardware pins for TMS570 board were configurable as many were not implemented within BSW Editor tool. But this was not too much of a hassle in this case, as code for this could easily be added within the actual source code for that module. But for large projects, support from ArcCore's team might be required.

2. Usability

   - *High-level GUI*
     Arctic Studio is built on Eclipse IDE and as a result provides all intuitive features like automatic code completion, syntax highlighting, integrated validation etc. It has a very familiar look and feel and as a result usability was not a problem.

   - *Re-usability, Modifiability and Intuitiveness*
     Arctic Studio includes all the above attributes within its tool. Since the tool is based on ARTOP platform and Eclipse IDE, it provides many benefits derived from these two base platforms. Although some parts like modeling can be further improved.

   - *Documentation*
     Most of the source files are well documented. User documentation from ArcCore [78] also provides enough information to get started with the tool. But information for implementing more advanced software features like integrating MCAL modules, multi-core implementation could be further improved.

3. Interoperability

   - *Data exchange and compliance to standard*
     Arctic Studio complies to AUTOSAR standard and standardized data exchange formats. But the tool can be further improved in this regard. For example, a dedicated authoring tool would make it easier to create ARXML, DBC, LDF, CDD files and others.

- *AUTOSAR interfaces, standard protocols and libraries*
  The tool complies to all the above requirements seamlessly.

- *Integration with third-party tools*
  Integrating Arctic Studio tools with third-party software is not a smooth or a straightforward approach. Source code is encrypted at many places and it is not possible to use the code generators from Arctic Studio and one should build their own code generator for each module. Although, this could be done, it is time consuming and error prone and hence not recommended.

4. Testability

   - *Debugging (Validation)*
     Arctic Studio tool offers validation feature to validate configured BSW modules in order to determine if the values entered are valid or not. There is no explicit debugger present in the tool which is another drawback.

   - *Simulation and Virtual Function Bus*
     There is no explicit simulation features available yet for early testing of the system. Although, there are timing extensions like VFB timing events provided by ARTOP platform but apart from this the tool itself does not provide any other features for testing the system at an early phase.

5. Service and support

   - *Migration support*
     ArcCore provides quick customer service and support required. It also provides migration support for companies to adopt AUTOSAR architecture within their software development processes.

   - *Workshops and customer care*
     The online user documentation provides a lot of information. Apart from that any other service could be arranged upon further consultation with the company.

6. Cost and distribution

   - *Tool-suite cost, duration of license, format and distribution time*
     ArcCore is one of the very few AUTOSAR companies that offers their tools and services for a trial period by which AUTOSAR is accessible to universities and students. But after tool comparisons, it can be concluded that though Arctic Studio and Arctic Core are full fledged tools, they are still not fully competitive when compared to other tool vendors like Vector Gmbh, as yet. Distribution of this tool is very simple and possible via registering on their website for getting the license. Upon obtaining the license key, the tool can be used for a given period of time and the license is updated each year.

   - *Market penetration, latest release and release interval*
     ArcCore is a one of the fastest growing companies in Sweden and one among the competitors in AUTOSAR market.

Figure 7.1 shows a comparison between before and after using Arctic Studio tool evaluated in terms of the selection criteria for this application. It can be seen that *functionality* and *usability* scores high after using the tool-chain. Although, it did not perform well in terms of *interoperability* and *testability* due to the challenges faced while implementing the demonstrator. Since the tool was procured on the basis of trial license not much support was provided although this might not be the case in the usual case.

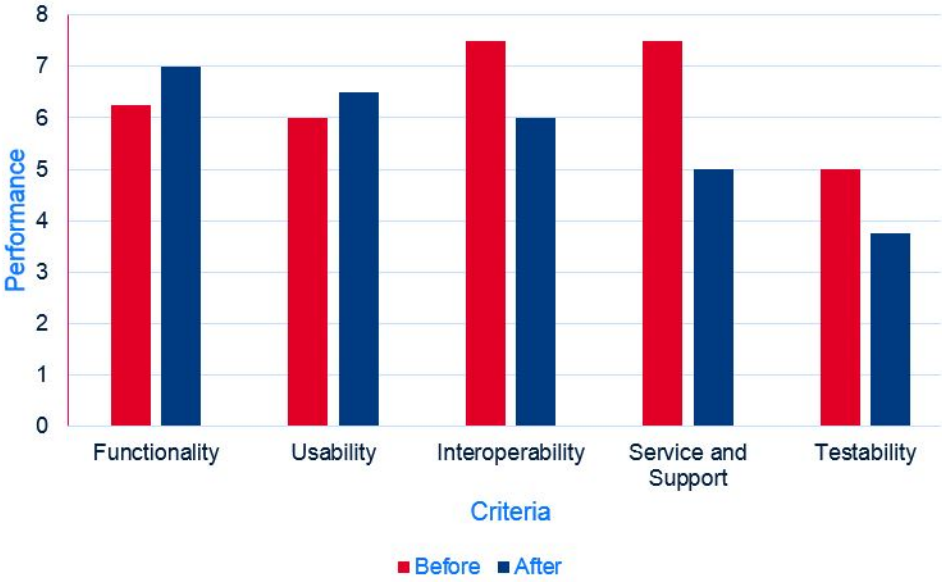


Figure 7.1: Arctic Studio tool evaluation based on documentation (Before) and practical experience (After)

7.2 Evaluation of AHP algorithm for selecting AUTOSAR tools

1. *Performance of AHP algorithm in structuring criteria into appropriate context and differentiating them from the alternatives*
   Using hierarchical framework of AHP algorithm, it is possible to clearly differentiate goals, sub-goals, criteria and alternatives from one another.

2. *Performance of AHP algorithm under various uncertainty and lack of enough data*
   AHP tool is used by decision makers to solve problems of choice under uncertainty or as a tool for prediction. With the help of weighted pairwise comparisons (a value ranging from 1 - 10), several co-efficients can be determined such as criteria comparison, comparing multiple alternatives and comparing uncertain events or scenarios in terms of probability of its realization for these factors. As a result, a prediction using AHP focuses on the distribution of relative probabilities of future outcomes.

7.2.1 Drawbacks of AHP algorithm

1. Pair-wise comparisons took a considerable amount of time. But this is because of four sub-goals in this application. As a result, the algorithm works better for a single goal while having the number of criteria and alternatives not more than 15.

2. The comparisons are based on the subjective opinion of the decision-maker. As a result adding or deleting alternatives from the list of initial comparisons will give a completely new result each time. This can be avoided by limiting uncertainty and gathering reliable data for respective alternatives.

Chapter 8

Conclusion

This chapter reflects on the research questions and objectives that were described in the introduction. It also reflects on the experiences gained with the tool chain from ArcCore and recommended practices, through the development of a simple yet illustrative AUTOSAR compliant system. Finally, this chapter is concluded with the scope for future work.

8.1 Reflection on thesis goals and objectives

The main goal of this thesis was to investigate and devise a tool selection methodology for selecting AUTOSAR tools based on a certain required criteria. In this case, the key criteria was to select a suitable *"low-cost"* AUTOSAR tool-chain and further evaluate this tool-chain by implementing AUTOSAR architecture on a hardware platform.

The first objective with regards to this goal was to select a suitable Multi Criteria Decision Making (MCDM) method which was further used to select an appropriate AUTOSAR tool-chain. Next, the stakeholders for using the tool-chain were identified and their key concerns and requirements were captured with the help of architectural UML / SysML models. Based on this gathered information, and with the acquired theoretical knowledge about the AUTOSAR architecture, a list of selection criteria was derived in order to select an AUTOSAR tool-chain. At this stage, the selected MCDM method called *"AHP"*, was applied for each AUTOSAR tool category. Overall, (i.e. considering the performance metric across all four tool categories), the top five AUTOSAR tool-chain (in the order of ranking) were from the following tool-vendors: AUTOSAR tool-suite by *Vector GmbH*, AUTOSAR Builder by *Dassault Systems*, *ETAS* AUTOSAR tool-chain, *Mentor Graphics* AUTOSAR tool-suite and tool suite from *ArcCore*.

When *low-cost* was considered as the key selection criteria, it resulted in selection of the following tools:

- Mathwork's Embedded Coder for modeling the system and the application layer.
- ArcCore's Arctic Studio tool for code generation purposes of application layer, BSW layer and RTE layer.
- COMASSO's BSW modules and configuration tool for configuration of the basic software modules.

Among the selected low-cost tools, we could only get access to the Arctic Studio tool-chain and as a result it was finally used to implement the demonstrator and was further evaluated (Chapter 7).

All objectives were met with satisfactory results thereby achieving the main thesis goal. Although, some significant challenges were faced while achieving the last objective (explained further), the final thesis goal could be met.

8.2 Reflection on research questions

1. **RQ1.1** Which MCDM method is likely to perform better for this application and why?
As discussed in Chapter 2, after researching other available MCDM methods, AHP algorithm was selected and was further put to work in this application to select a suitable AUTOSAR tool-chain. Main reasons why AHP method stood out from the rest were its distinctive hierarchical framework, pair-wise comparison concept, feasibility and conformance to the required project size.

2. **RQ2.1** Who are the key stakeholders in development of an automotive ECU using AUTOSAR architecture?
In this case, the key stakeholders were identified upon discussions with Engineers from Brace Automotive and Orlaco. Furthermore, the stakeholders considered were in perspective of users who *use the AUTOSAR tool-chain* to develop an AUTOSAR ECU. The key stakeholders identified were requirement analyst, system designer, software designer, application developer, system developer and system engineer. Requirement analyst mainly does the task of gathering the customer requirements of the system and analyzing the requirements in order to determine the most important ones to get the development process started. This person works closely with software and system designer. Application developer and system developer develop the required functionalities of the system while the system engineer integrates the modules. To aid all these development processes, each of the stakeholders make use of specific AUTOSAR tool-chain and gathering their requirements to select an appropriate AUTOSAR tool-chain was the main intent. This research question is further explained in depth in Chapter 4.

3. **RQ2.2** What are their key concerns, requirements and constraints in terms of using AUTOSAR tool-chain to develop automotive ECUs seamlessly?
The key requirements for each of the development process mentioned above were captured in Figure 4.3 in Chapter 4. At first, four tool categories were identified based on the AUTOSAR methodology. Further, requirements were fine tuned with the help of the acquired AUTOSAR knowledge and conversations with a team of experienced engineers. The requirements diagram also helped in understanding the tool dependencies (e.g. RTE tool requires artifacts / output from other tools) which further gave a clear picture in understanding the control flow of the entire process.

4. **RQ3.1** What are the different architectural views and models that addresses the concerns of stakeholders?
Since the main task here was to identify the requirements from the standpoint of stakeholders who *use* these tools, *logical architecture view* was considered. This process gives a clear picture of how the stakeholders interact with the system (in this case, the term system is used in the context of AUTOSAR tool-chain). As a result, use-case diagrams, as shown in Section A.1 (appendix A) was put forth to understand how each stakeholder interacts with the system. Further, with the help of this diagram, it was also possible to back-trace the requirements for a particular use-case, which in turn helped in understanding the purpose of a particular requirement.

5. **RQ3.2** How to derive the key criteria for selecting the tool-chain from these models?
With the help of these models, many ambiguities could be addressed. As a result, the main requirements for each of these tools were clear, to a certain extent. These identified requirements were categorized into 6 top-level criteria namely functionality, usability, interoperability, service & support, cost & distribution and testability. Each of these top-level criteria were further fine tuned to derive specific qualifying parameters.

6. **RQ3.3** What are the different criteria that are finally selected in order to opt for an AUTOSAR tool-chain?
The hierarchical framework as shown in Figure 4.10 provides a clear picture of the criteria

and sub-criteria that were finally selected. It also gives a list of AUTOSAR tool alternatives which were considered for ranking upon the application of AHP algorithm.

7. **RQ4.1** Which tool outperforms others in terms of performance as the key selection criteria?
As shown in Figure 4.15, the tool-sutie provided by Vector GmbH, Dassault Systems, ETAS, Mentor Graphics and ArcCore were selected as the top 5 performing tools, in that order. More detailed graphs relating to how each tool performs under each criteria can be found in Section A.2 (Appendix A).

8. **RQ4.2** Which tool is better when cost is considered as a key criteria for selecting AUTOSAR tool-chain?
When cost was considered as a key selection criteria, a tool-suite provided by a single tool vendor could not be found. As a result, tools were ranked and opted for each tool category. Embedded Coder from Mathworks outperformed others among modeling tools, ArcCore's Arctic Studio was selected as the best low cost tool among BSW and RTE tools. Further, COMASSO outranked others for obtaining BSW modules and BSW configuration tool, as shown in the graphs under Section A.3. The only issue in selecting tools from different vendors is that the tools don't integrate seamlessly and manual integration might be required and therefore it is better to minimize the use of tools from different tool vendors.

9. **RQ4.3** What are the trade-offs made in answering the question RQ4.2?
In order to rank tools based on low-cost, some compromises had to be made in terms of usability, testability, interoperability, service & support and other value added services like virtual ECU, use of functional safety standards etc., that potentially increases the cost of a tool-chain. As mentioned above, integrating tools from multiple tool vendors could further increase the overhead.

10. **RQ5.1** Did the selected low-cost tool-chain perform well when applied to the demonstrator?
Initial implementation of AUTOSAR architecture using the selected Arctic Studio tool-suite on an unsupported hardware platform (Arduino Mega 2560) by the tool, led to an understanding that it is critical for opting an hardware platform which is supported by the tool being used. The main challenge here is, integrating manual code within the code generators. During the final implementation, the tool worked seamlessly when opted for an hardware platform (TMS570LC4357) that was supported by Arctic Studio.

11. **RQ5.2** What were the various challenges faced in this regard?
The challenges faced are as listed in Section 5.1.4. Fundamentally, integration of manual code was the biggest challenge.
Moreover, in this thesis, two different goals were combined in search of a low-cost tool, i.e. to opt for a tool which is not too expensive and using a low-powered micro-controller unit for implementation. But in this experience, combining these two goals was not feasible. For instance, if one prefers to run an AUTOSAR application on a small micro-controller then Vector GmbH offers Microsar operating system which is specifically devised for implementing AUTOSAR for low-powered micro-controllers. But the Microsar tool is not a low-cost option.

12. **RQ5.3** How did the selected MCDM tool perform? Are there any recommendations to improve the selection criteria or methodology to select the AUTOSAR tool-chain?
AHP tool performed well for this application, although comparisons took a considerable amount of time. This was because there were four sub-goals and the number of comparisons to be made multiplied four times. But this is true for any other decision making method. By decreasing the number of alternatives to be compared even further (say not more than 10 tools) would further increase the performance of the proposed tool selection methodology (i.e. optimizes the time taken for each pair-wise comparison made).

8.3 Reflection on research methodology

The main goal of this thesis was to develop a methodology for selecting AUTOSAR tool-chain having cost as a key criteria. With regards to this, the overall research methodology suited *satisfactorily* for this thesis. A literature study on the available MCDM tools and theoretical pre-study of AUTOSAR formed a solid base for implementing AUTOSAR architecture. Unfortunately, this approach lacked a practical pre-study and understanding of the use of different AUTOSAR tools and their bottlenecks. One insight gained during the actual implementation is the dependency between the selected hardware platform and software tools. The decision to use the Arduino Mega 2560 hardware without assessing the risks involved in using the Arctic Studio tool on a platform with limited resources and lack of right support at that stage proved challenging. Apart from this setback, the entire process of implementing AUTOSAR architecture on TMS570 board did not lead to any significant bottlenecks. From this experience, it is worthwhile to point out that although AUTOSAR as a standard is supposed to be hardware independent, implementing the same using a tool is not a straightforward approach and support of AUTOSAR tools for the underlying hardware platform is highly recommended.

8.4 Recommended practices

1. It is important to select the hardware that the tool supports. Even-though AUTOSAR standard specifications offer standard APIs, which are implementable on any hardware in theory, it is not that straight forward in reality. It is not feasible to integrate a new hardware within an existing tool without a lot of manual modifications which defies the purpose of using a tool. Besides, for such an integration, it is recommended to invest in workshops and support from the tool vendors throughout the project life cycle.

2. For low-cost tool alternatives some key insights are:

   - Investing in the right tools for implementing AUTOSAR is a crucial step. These tools are expensive and hence initial investment required for adopting AUTOSAR is huge but it could be a strategic decision and should be seen as investing in the future.
   - Having said that, considering third party integration companies could be another option. For example, Embitel provides services to integrate 3rd party MCAL modules and BSW layer from Comasso within the Vector tool-suite. But this option was not tested in this thesis, hence not much can be commented on reducing the overall cost.

8.5 Future work

This thesis can be further extended to evaluate the Arctic Studio tool by testing it on multicore ECUs, spanning across more than one hardware unit. Also, graphical modeling of the AUTOSAR application layer can be explored instead of using the ARText language to model the application components.

Bibliography

[1] R.N. Charette. IEEE Spectrum: Technology, Engineering, and Science News. (2018). This Car Runs on Code.
`https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code` [Accessed: Jan. 20, 2017]. 1

[2] Simon. Fürst, Challenges in the Design of Automotive Software. BMW Group. Munich, Germany, 2010.
`https://www.date-conference.com/proceedings-archive/PAPERS/2010/DATE10/PDFFILES/03.8_1.PDF.` [Accessed: Jan. 20, 2017]. 1

[3] M. Pesce, "Software Takes On More Tasks in Todays Cars," in Autopia, WIRED, 2011.
`https://www.wired.com/2011/04/the-growing-role-of-software-in-our-cars/.` [Feb. 1, 2017]. 1

[4] N. Tracey, U. Lefarth, H. Wolff, U. Freund. ETAS GmbH, Stuttgart. ECU Software Module Development Process Changes in AUTOSAR.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.8004&rep=rep1&type=pdf. [Accessed: Jan. 20, 2017]. 1

[5] B. Jungk.
`"Automotive Security State of the Art and Future Challenges"`. Physical Analysis and Cryptographic Engineering (PACE). Temasek Laboratories at Nayang Technological University, Singapore. 1

[6] N. Tracey. U. Lefarth. H.J. Wolff. U. Freund.
`"ECU Software Module Development Process Changes in AUTOSAR "`. ETAS GmbH, Germany. 1

[7] N. Navet, F. Simonot-Lion. `"Automotive Embedded Systems Handbook"`, ISBN -13: 978-0-8493-8026-6, 2009. 2

[8] AUTOSAR, "AUTOSAR: Background," 2016.
`http://www.autosar.org/about/basics/background/.` [Accessed: Jan. 20, 2017.]. 2

[9] AUTOSAR, "Core Partners", 2016.
`https://www.autosar.org/partners/current-partners/core-partners/.` [Accessed: May. 1, 2017.]. 2

[10] B. Schtz, A. Vallecillo, and P. Clarke, Model driven engineering languages and systems: 16th international conference, models 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings, A. Moreira, B. Schatz, and J. Gray, Eds. Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2013. In *Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem?*, page 10, 2013. 3

[11] V. I. GmbH, "Vector AUTOSAR tools", 2010.
`https://vector.com/vi_ autosar_ tools_ en.html.` [Accessed: Jan. 18, 2017]. 3, 27

[12] Elektrobit. AUTOSAR - Elektrobit, 2018.
https://www.elektrobit.com/products/ecu/technologies/autosar/. [Accessed: Jan. 5, 2018]. 3, 26

[13] ETAS. ETAS AUTOSAR Solutions, AUTOSAR Applications, ETAS Products.
https://www.etas.com/en/products/applications_ autosar.php. [Accessed: Jan. 5, 2018]. 26

[14] AUTOSAR, ”Premium Partners”, 2016.
https://www.autosar.org/partners/current-partners/premium-partners/. [Accessed: May. 1, 2017.]. 2, 3

[15] S. Waldron. "Introduction to AUTOSAR". Vector GB. 2015. 2

[16] Analytic Hierarchy Process & Making Key Business Decisions.
https://www.handshake.com/blog/analytic-hierarchy-process-2/. [Accessed: Jan. 14, 2018.].

[17] AUTOSAR."Interoperability of AUTOSAR". AUTOSAR Release 4.2.2. 2016. 3, 9, 35

[18] Agile Business Consortium. DSDM Atern Handbook.
https://www.agilebusiness.org/content/moscow-prioritisation-0. 2008. [Accessed: May. 1, 2017.].

[19] M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, D. Ratiu.
"Seamless Model-Based Development : From Isolated Tools to Integrated Model Engineering Environments". IEEE, Vol. 98, No. 4, April 2010. 7, 8

[20] J. Holtmann, J. Meyer, M. Meyer. s-lab Software Quality Lab, Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn.
"A Seamless Model-Based Development Process for Automotive Systems". 2011. 7

[21] C.V. Ramamoorthy, C. Chandra, H.G. Kim, Y.C. Shim, V. Vij. University of California, Berkeley.
"Systems Integration: Problems and Approaches". s-lab Software Quality Lab, Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn. 7

[22] R.F. Boldt. IBM Software Group.
"Modeling AUTOSAR systems with a UML/SysML profile". Automotive electronics and software development. White paper. July 2009. 8

[23] M. Pagel, M. Brorkens. BMW AG, Carmeq GmbH.
"Definition and Generation of Data Exchange Formats in AUTOSAR". lNCS 4066, pages pp. 5265, 2006. 8

[24] S. Voget.
"SAFE RTP: An open source reference tool platform for the safety modeling and analysis". Embedded Real Time Software and Systems Conference Proceedings, 2014. 8

[25] M. Rudorfer, S. Voget, S. Eberle.
"Artop (AUTOSAR Tool Platform)". Whitepaper. March 2009. 8, 24

[26] G. Sandmann, R. Thompson.
"Development of AUTOSAR Software Components within Model-Based Design". The MathWorks, Inc. 2008-01-0383.

[27] J.S. Hammond, R.L. Keeney, H. Raiffa.
"Fatta smarta beslut". Forma Books AB, 2001. 8

[28] J.S. Dyer, P.C. Fishburn. R.E. Steuer, J. Wallenius, S. Zionts. "Multiple Criteria Decision Making, Multiattribute Utility Theory: The Next Ten Years". Management of Science, Vol 38, No. 5. May 1992. 11

[29] T.L. Saaty. Katz Graduate School of Business, University of Pittsburgh. "Decision making with the analytic hierarchy process ". Int. J. Services Sciences, Vol. 1, No. 1, 2008. 9, 44

[30] E. Triantaphyllou, S.H. Mann. "Using The Analytic Hierarchy Process For Decision Making In Engineering Applications: Some Challenges". Interl Journal of Industrial Engineering: Applications and Practice. Vol. 2, No. 1, pp. 35-44, 1995. 44

[31] N. Kadoic, N.B.Redep, B. Divjak. Faculty of Organisation and Informatics. Croatia. "Decision Making With The Analytic Network Process". 9

[32] K. Govindan, M. B. Jepsen. "ELECTRE: A comprehensive literature review on methodologies and applications". European Journal of Operational Research 250 (2016) 1-29, ELSEVIER. 10

[33] M. Behzadian, R.B. Kazemzadeh, A. Albadvi, M.Aghdasi. "PROMETHEE: A comprehensive literature review on methodologies and applications". European Journal of Operational Research 200 (2010) 198-215, ELSEVIER. 11

[34] X.S. Qin, G.H. Huang, A. Chakma, X.H. Nie, Q.G. Lin. "A MCDM-based expert system for climate-change impact assessment and adaptation planning A case study for the Georgia Basin, Canada". European Journal of Operational Research 200 (2010) 198-215, Expert Systems with Applications, 34(3): 2164-2179. 11

[35] AUTOSAR."Requirements on Interoperability of Autosar Tools". AUTOSAR Release 4.2.2. 2016.

[36] C.T. Kuah, K.Y. Wong, F. Behrouzi."A Review on Data Envelopment Analysis (DEA)". Fourth Asia International Conference on Mathematical / Analytical Modelling and Computer Simulation. 2010. 10

[37] L.A. Zadeh."Fuzzy Sets*". University of California, Berkeley, California. INFORMATION AND CONTROL 8, 338-353 (1965). 10

[38] M. Tamiz, D.F. Jones, E. El-Darzi."A review of Goal Programming and its applications". England. Annals of Operations Research 58(1995)39-53. 10

[39] Expert Choice. 1990. https://expertchoice.com/.[Accessed: Jan. 4, 2018.].

[40] AUTOSAR, "AUTOSAR: Layered Software Architecture.", AUTOSAR Release 4.2.2. http://www.autosar.org/fileadmin/files/standards/classic/4-2/software-architecture/general/auxiliary/AUTOSAR _EXP_LayeredSoftwareArchitecture.pdf. [Accessed: Jan. 24, 2017]. 15, 20

[41] R. Hebig, Hasso Plattner Institut. Methodology and Templates in AUTOSAR. 21

[42] AUTOSAR, "AUTOSAR: Partners," 2017. https://www.autosar.org/partners/current-partners/. [Accessed: Jan. 20, 2017.].

[43] AUTOSAR, "Basic Software", 2016.
`https://www.autosar.org/about/technical-overview/ecu-software-architecture/autosar-basic-softw`
[Accessed: May. 1, 2017.].

[44] Controller Area Network (CAN) Overview, Controller Area Network (CAN) Overview - National Instruments. [Online].
`http://www.ni.com/white-paper/2732/en/`. [Accessed: May. 1, 2017].

[45] S.M.Fernndez, C.P.Ayala, X.Franch. Universitat Politcnica de Catalunya, Barcelona, Spain. E.Y.Nakagawa. `"A Survey on the Benefits and Drawbacks of AUTOSAR"`, University of So Paulo, So Carlos, Brazil. 2015.

[46] ArcCore AB, Gothenburg, Sweden. "ArcticCore, ArcticStudio".
`https://www.arccore.com/products/`. [Accessed: June. 09, 2017.].

[47] IBM Rhapsody, USA. "Rational Rhapsody Designers for System Engineers".
`http://www-03.ibm.com/software/products/en/ratirhapdesiforsystengi/`. [Accessed: June. 09, 2017.].

[48] Comasso e.V, Germany. "Comasso".
`https://www.comasso.org/`. [Accessed: June. 09, 2017.]. 25

[49] S. Corrigen. `"Introduction to the Controller Area Network (CAN)"`. Texas Instruments, Application Report. May 2016.

[50] "Introduction to the Controller Area Network (CAN)," Texas Instruments. Application Report, 2002.
`http://www.ti.com/lit/an/sloa101a/sloa101a.pdf`. [Accessed: Jan. 24, 2017].

[51] N.R. Kandimala, M. Sojka, Czech Technical University, Czech Republic.
`Safety and Security Features in AUTOSAR.`, Thursday 15th November, 2012.

[52] AUTOSAR, "AUTOSAR: Utilization of Crypto Services.",
`http://www.autosar.org/fileadmin/files/standards/`
`classic/4-2/software-architecture/safety-and-security/`
`auxiliary/AUTOSAR_EXP_UtilizationOfCryptoServices.pdf`. [Accessed: Jan. 24, 2017].

[53] R. Kazman, L. Bass, G. Abowd, M. Webb.
`"SAAM: A method for analyzing the properties of software architectures"`. Proceedings of the 16th International Conference on Software Engineering, pp. 81-90 (IEEE Computer Society, Sorrento, Italy, 1994).

[54] R. Kazman, M. Klein, P. Clements.
`"ATAM: Method for architecture evaluation"`. CMU/SEI-2000-TR-004. Carnegie Mellon University, Pittsburgh, Pennsylvania. 2000.

[55] R. Kazman, J. Asundi, M. Klein.
`"Quantifying the costs and benefits of architectural decisions"`. Proceedings of the 23rd International Conference on Software Engineering, pp. 297-306. IEEE Computer Society, Toronto, Ontario, Canada. 2001.

[56] "Arccore - For Application Developers".
`https://www.arccore.com/products/arctic-studio/for-application-developers` [Accessed: Jan. 24, 2017]. 24

[57] "Arccore - For Platform Developers".
`https://www.arccore.com/products/arctic-studio/for-platform-developers` [Accessed: Jan. 24, 2017]. 24

[58] ”Arccore - For Integrators”. https://www.arccore.com/products/arctic-studio/for-integrators [Accessed: Jan. 24, 2017]. 24

[59] A. Graf. ”Support for vendor specific parameter / module definitions in COMASSO basic software configuration tool”. 5ise. http://5ise.quanxinquanyi.de/2013/11/15/support-for-vendor-specific-parameter-module-definitio [Accessed: Jan. 24, 2018]. 25

[60] A. Graf. ”COMASSO BSW generation and validation”. 5ise. http://5ise.quanxinquanyi.de/2013/11/04/comasso-bsw-generation-and-validation/comment-page-1/ [Accessed: Jan. 24, 2018]. 25

[61] S. Anssi. S. Gerard. S. Kuntz. F. Terrier. AUTOSAR vs MARTE for Enabling Timing Analysis of Automotive Applications. p-272. SDL 2011: Integrating System and Software Modeling.". 15th International SDL Forum Toulouse, France. July 5-7, 2011. 25

[62] A. Junghanns. J. Mauss. M. Seibt. Faster Development of AUTOSAR compliant ECUs through simulation. ERTS - Embedded Real Time Software and Systems, Toulouse. 05 - 07.02.2014. 25

[63] ”Dassault Systemes. AUTOSAR Builder - AUTOSAR Applications and Systemes”. https://www.3ds.com/products-services/catia/products/autosarbuilder/. [Accessed: Jan. 24, 2018]. 25

[64] ”dSPACE. System Desk - Modeling system architecture and generating virtual ECUs”. https://www.dspace.com/en/pub/home/products/sw/system_ architecture_ software/systemdesk.cfm/. [Accessed: Jan. 24, 2018]. 25

[65] ”dSPACE. Target Link - Production code generation for the highest demands”. https://www.dspace.com/en/pub/home/products/sw/pcgs/targetli.cfm. [Accessed: Jan. 24, 2018]. 25

[66] S. Voget. P. Favrais. "How the concepts of the Automotive standard "AUTOSAR" are realized in new seamless tool-chains". Continental Engineering Services GmbH. 25

[67] A. Graf. M. Volter. "Integrating the AUTOSAR tool chain with Eclipse based model transformations". Itemis GmbH, Germany. 24

[68] K. Hoffmeister. "Automated Driving Necessary Infrastructure Shift". Elektrobit, Germany. 26

[69] KPIT. "AUTOSAR Handbook". KPIT Technologies Ltd. 26

[70] Embedded Coder. ”Generate C and C++ code optimized for embedded systems”. Mathworks. https://www.mathworks.com/products/embedded-coder.html [Accessed: Jan. 24, 2018]. 26

[71] Product Flyer COQOS SDK. "https://www.opensynergy.com/fileadmin/user_ upload/Datenblaetter/Datasheet_ COQOS.pdf". Opensynergy. 27

[72] C. Hammerschmidt. ”Mentor Graphics, Mecel offer Autosar 4.x software design solution”. http://www.eenewseurope.com/news/mentor-graphics-mecel-offer-autosar-4x-software-design-soluti [Accessed: Jan. 24, 2018]. 27

[73] Mentor Graphics. ”Volcano Architecture”. https://www.mentor.com/embedded-software/volcano-automotive/. [Accessed: Jan. 24, 2018]. 27

[74] Systems and Software Engineering. ”Architecture description”. ISO/IEC/IEEE 42010. http://www.iso-architecture.org/ieee-1471/defining-architecture.html. [July, 24, 2017]. 41

[75] Arduino, ”ArduinoBoardMega2560,” 2017. https://www.arduino.cc/en/Main/arduinoBoardMega2560. [Accessed: Jan. 18, 2017].

[76] Atmel Corporation. Atmel ATmega640/1280/1281/2560/2561 datasheet. 2549QAVR02/2014. 55

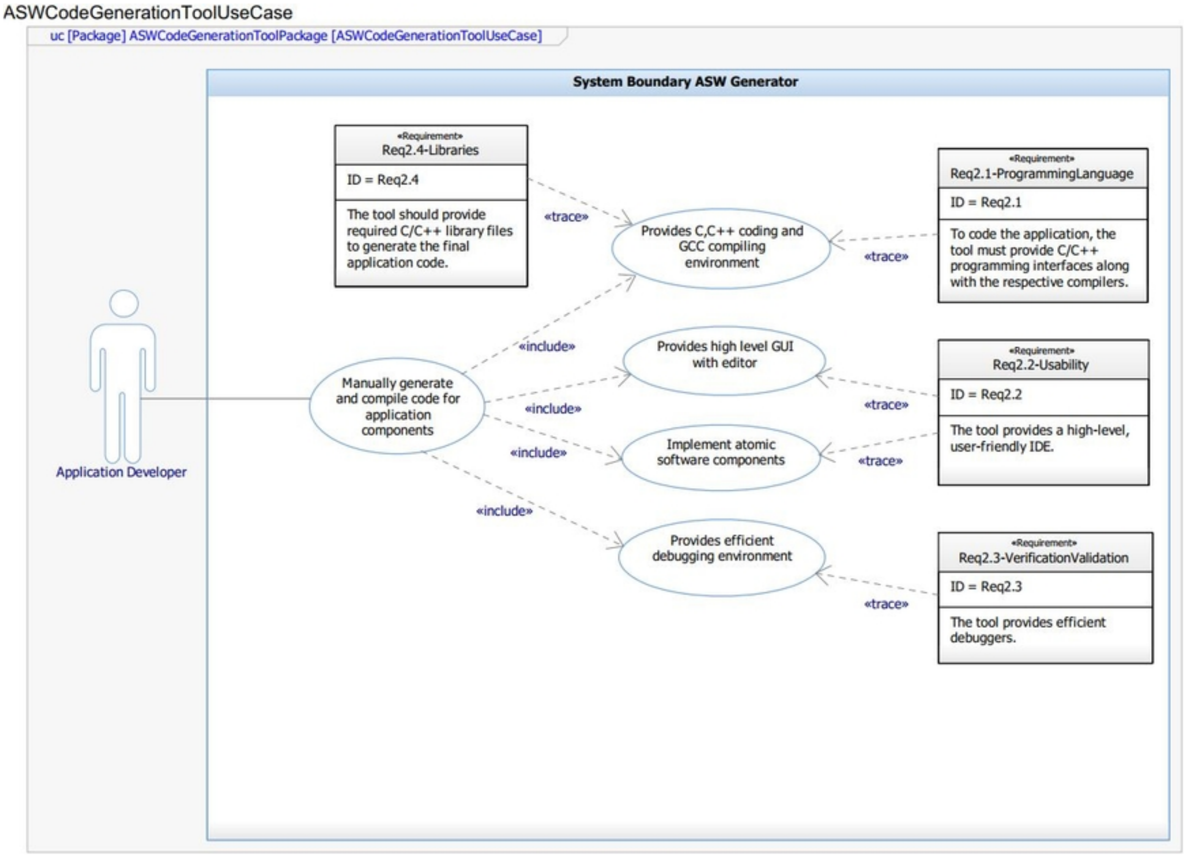
[77] Arduino, ”ArduinoBoardMega2560,” 2017. https://www.orlaco.com/product/orlaco-emos-ethernet-digital-hd-camera. [Accessed: Jan. 18, 2017]. 55

[78] ArcCore, ”ArcCore’s User Documentation”. https://www.arccore.com/my-arccore/documentation. [Accessed: Jan. 18, 2017]. 78

Appendix A

Tool selection methodology results

A.1 Tool use-case diagrams


ASWCodeGenerationToolUseCase
uc [Package] ASWCodeGenerationToolPackage [ASWCodeGenerationToolUseCase]
System Boundary ASW Generator
«Requirement»
Req2.4-Libraries
ID = Req2.4
«Requirement»
Req2.1-ProgrammingLanguage
ID = Req2.1
«trace»
Provides C,C++ coding and GCC compiling environment
«trace»
«include»
Provides high level GUI with editor
«Requirement»
Req2.2-Usability
ID = Req2.2
Manually generate and compile code for application components
«include»
«trace»
Implement atomic software components
«include»
«trace»
Application Developer
«include»
Provides efficient debugging environment
«Requirement»
Req2.3-VerificationValidation
ID = Req2.3
«trace»

The tool should provide required C/C++ library files to generate the final application code.

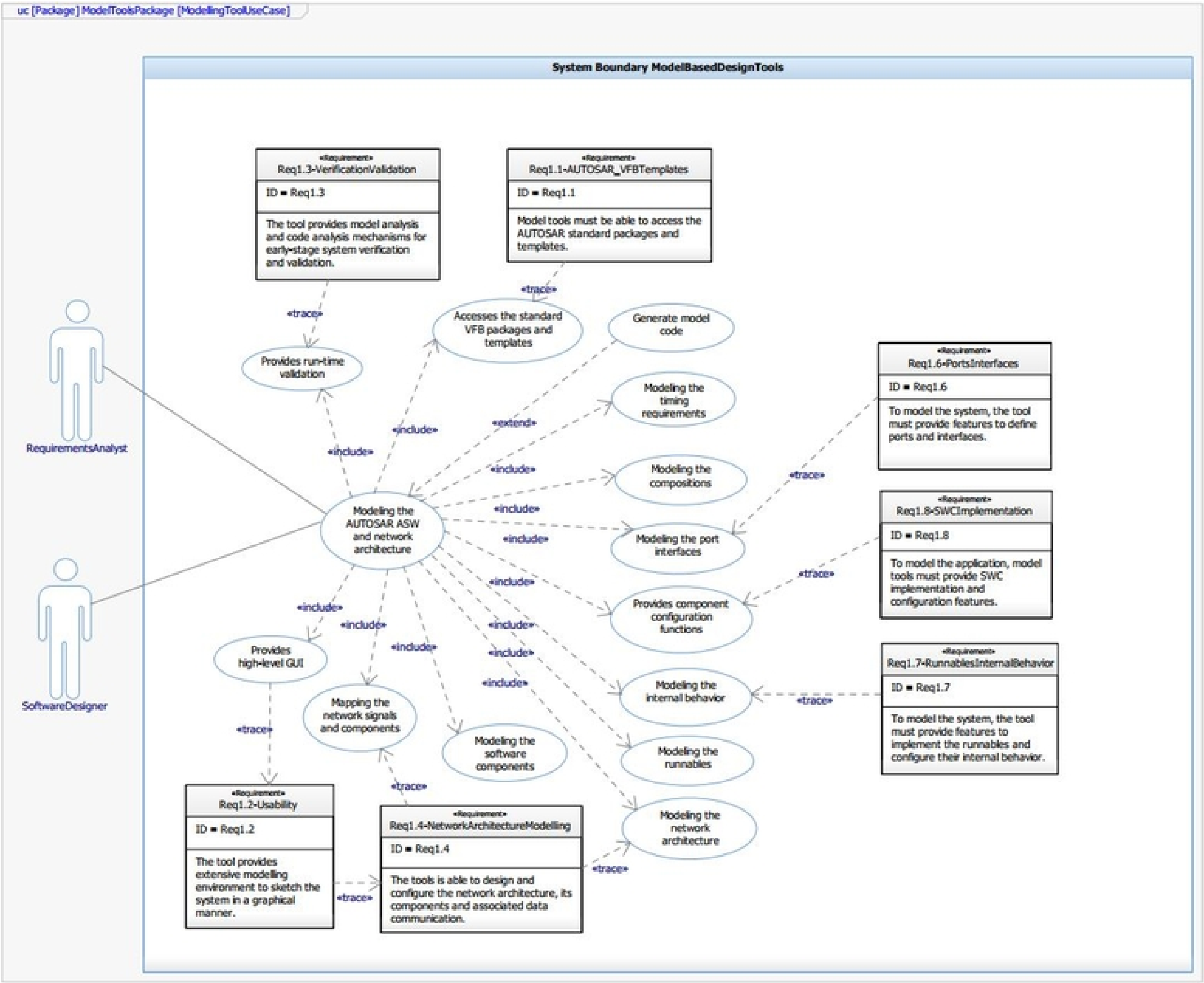
To code the application, the tool must provide C/C++ programming interfaces along with the respective compilers.

The tool provides a high-level, user-friendly IDE.

The tool provides efficient debuggers.

Figure A.1: Application Software tool use case diagram

<trace> relationship provides a general purpose relationship between a requirement and any other model element. In this case, trace is used to relate requirements which lead to a specific use case, hence when the use case (tool functionality) changes then the requirement also gets updated. This enables the traceability of the requirements via use cases.


uc [Package] ModelToolsPackage [ModellingToolUseCase]
System Boundary ModelBasedDesignTools
RequirementsAnalyst
SoftwareDesigner
«Requirement»
Req1.3-VerificationValidation
ID = Req1.3
«Requirement»
Req1.1-AUTOSAR_VFBTemplates
ID = Req1.1
«Requirement»
Req1.6-PortsInterfaces
ID = Req1.6
«Requirement»
Req1.8-SWCImplementation
ID = Req1.8
«Requirement»
Req1.7-RunnablesInternalBehavior
ID = Req1.7
«Requirement»
Req1.2-Usability
ID = Req1.2
«Requirement»
Req1.4-NetworkArchitectureModelling
ID = Req1.4
Provides run-time validation
Accesses the standard VFB packages and templates
Generate model code
Modeling the timing requirements
Modeling the compositions
Modeling the port interfaces
Provides component configuration functions
Modeling the internal behavior
Modeling the runnables
Modeling the network architecture
Modeling the AUTOSAR ASW and network architecture
Provides high-level GUI
Mapping the network signals and components
Modeling the software components
«trace»
«include»
«extend»

The tool provides model analysis and code analysis mechanisms for early-stage system verification and validation.

Model tools must be able to access the AUTOSAR standard packages and templates.

To model the system, the tool must provide features to define ports and interfaces.

To model the application, model tools must provide SWC implementation and configuration features.

To model the system, the tool must provide features to implement the runnables and configure their internal behavior.

The tool provides extensive modelling environment to sketch the system in a graphical manner.

The tools is able to design and configure the network architecture, its components and associated data communication.

Figure A.2: Modeling tools use case diagram

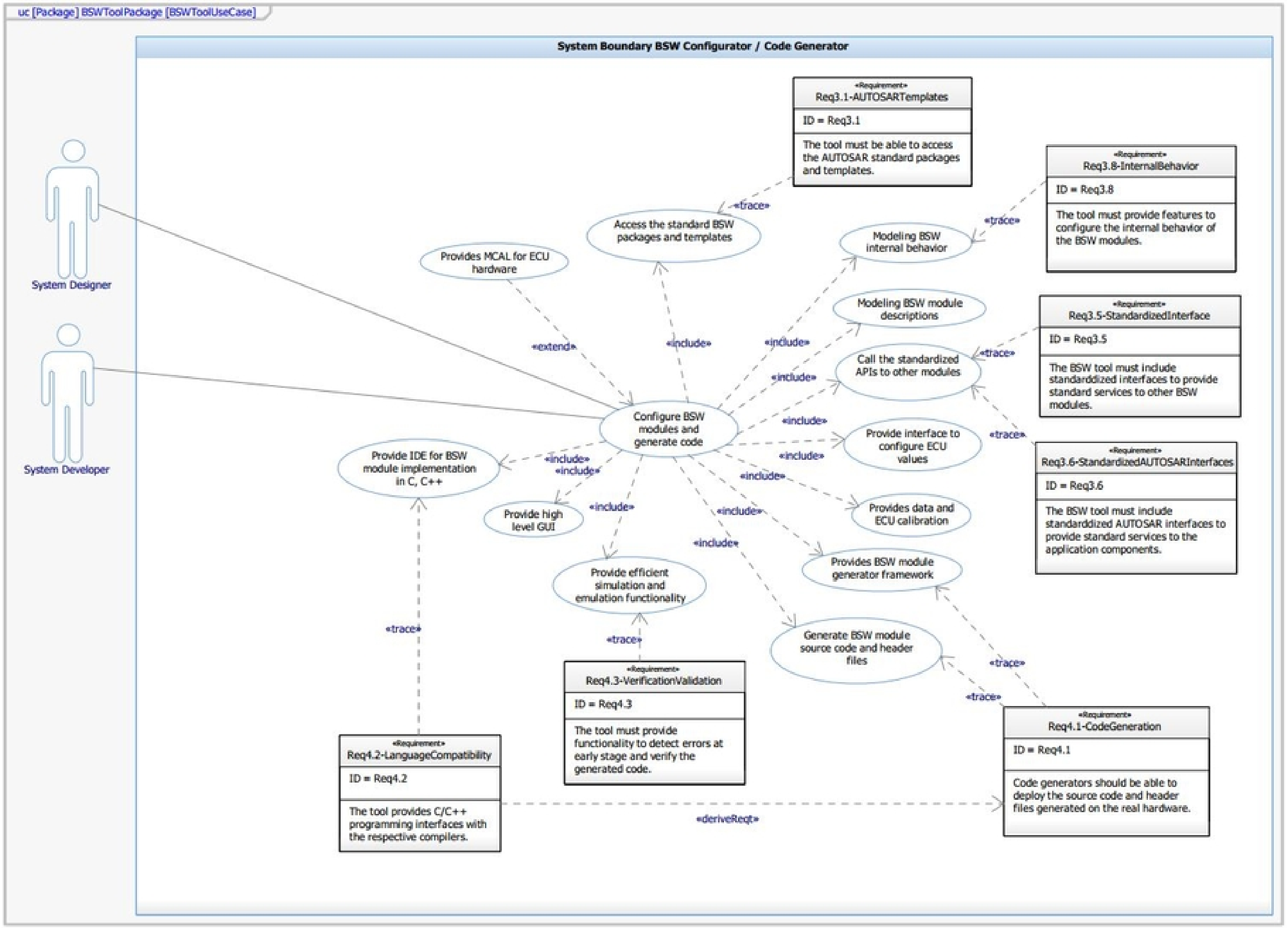

uc [Package] BSWToolPackage [BSWToolUseCase]
System Boundary BSW Configurator / Code Generator
System Designer
System Developer
«Requirement»
Req3.1-AUTOSARTemplates
ID = Req3.1
The tool must be able to access the AUTOSAR standard packages and templates.
«Requirement»
Req3.8-InternalBehavior
ID = Req3.8
The tool must provide features to configure the internal behavior of the BSW modules.
«Requirement»
Req3.5-StandardizedInterface
ID = Req3.5
The BSW tool must include standarddized interfaces to provide standard services to other BSW modules.
«Requirement»
Req3.6-StandardizedAUTOSARInterfaces
ID = Req3.6
The BSW tool must include standarddized AUTOSAR interfaces to provide standard services to the application components.
«Requirement»
Req4.3-VerificationValidation
ID = Req4.3
The tool must provide functionality to detect errors at early stage and verify the generated code.
«Requirement»
Req4.2-LanguageCompatibility
ID = Req4.2
The tool provides C/C++ programming interfaces with the respective compilers.
«Requirement»
Req4.1-CodeGeneration
ID = Req4.1
Code generators should be able to deploy the source code and header files generated on the real hardware.
Access the standard BSW packages and templates
Provides MCAL for ECU hardware
Modeling BSW internal behavior
Modeling BSW module descriptions
Call the standardized APIs to other modules
Configure BSW modules and generate code
Provide interface to configure ECU values
Provides data and ECU calibration
Provide IDE for BSW module implementation in C, C++
Provide high level GUI
Provide efficient simulation and emulation functionality
Provides BSW module generator framework
Generate BSW module source code and header files
«trace»
«extend»
«include»
«include»
«include»
«trace»
«include»
«trace»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«trace»
«trace»
«trace»
«trace»
«trace»
«deriveReqt»

Figure A.3: Basic software tool use case diagram

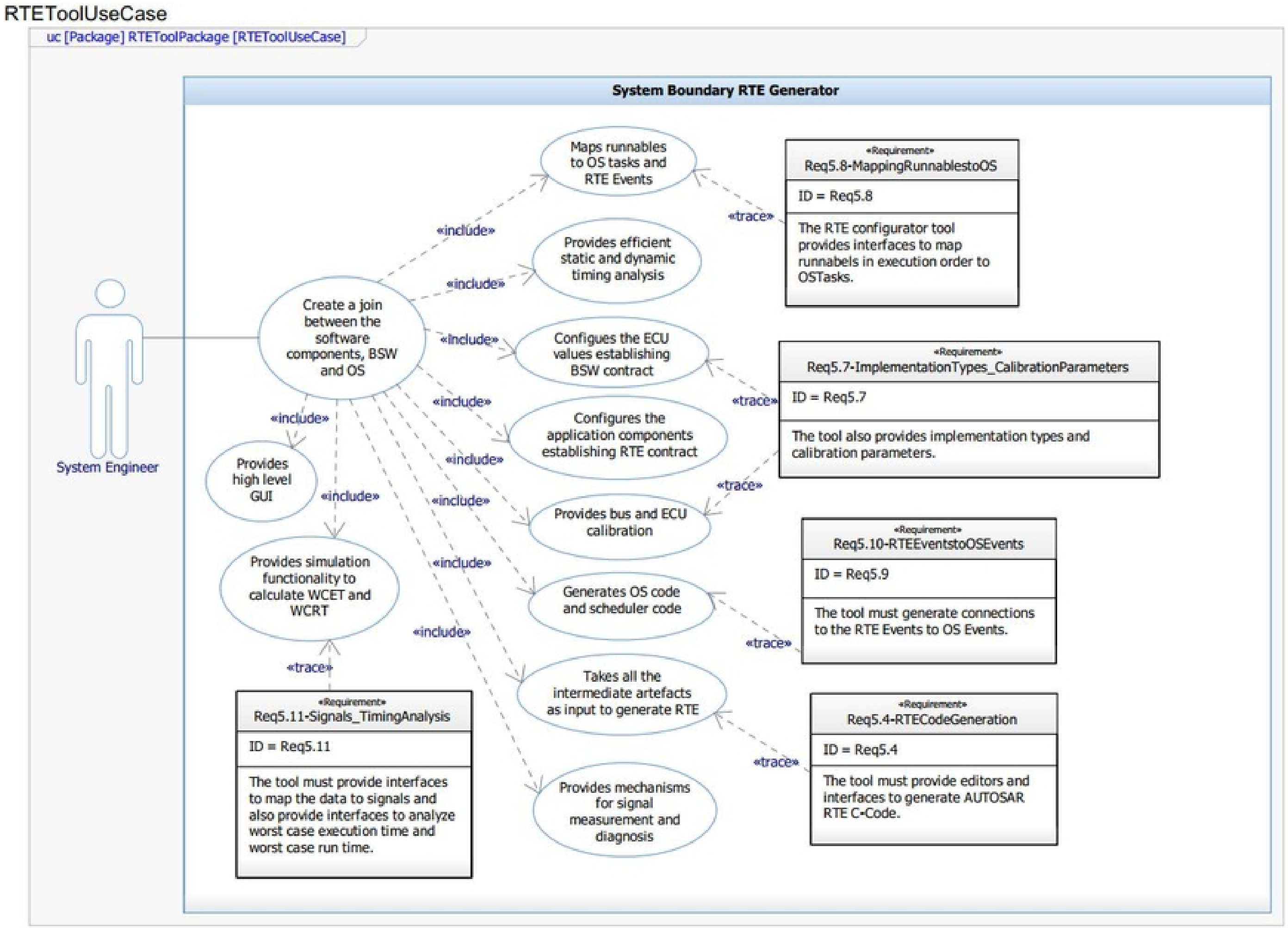

RTEToolUseCase
uc [Package] RTEToolPackage [RTEToolUseCase]
System Boundary RTE Generator
System Engineer
Create a join between the software components, BSW and OS
«include»
Maps runnables to OS tasks and RTE Events
Provides efficient static and dynamic timing analysis
Configues the ECU values establishing BSW contract
Configures the application components establishing RTE contract
Provides bus and ECU calibration
Generates OS code and scheduler code
Takes all the intermediate artefacts as input to generate RTE
Provides mechanisms for signal measurement and diagnosis
Provides high level GUI
Provides simulation functionality to calculate WCET and WCRT
«trace»
«Requirement»
Req5.8-MappingRunnablestoOS
ID = Req5.8
The RTE configurator tool provides interfaces to map runnabels in execution order to OSTasks.
«Requirement»
Req5.7-ImplementationTypes_CalibrationParameters
ID = Req5.7
The tool also provides implementation types and calibration parameters.
«Requirement»
Req5.10-RTEEventstoOSEvents
ID = Req5.9
The tool must generate connections to the RTE Events to OS Events.
«Requirement»
Req5.4-RTECodeGeneration
ID = Req5.4
The tool must provide editors and interfaces to generate AUTOSAR RTE C-Code.
«Requirement»
Req5.11-Signals_TimingAnalysis
ID = Req5.11
The tool must provide interfaces to map the data to signals and also provide interfaces to analyze worst case execution time and worst case run time.

Figure A.4: RTE tool use case diagram

A.2 Application of AHP - Results

A.2.1 Overall synthesis results for each tool category

The following synthesis graphs are w.r.t sub-goals (refer Figure 4.10 in Chapter 4).

Model Tools

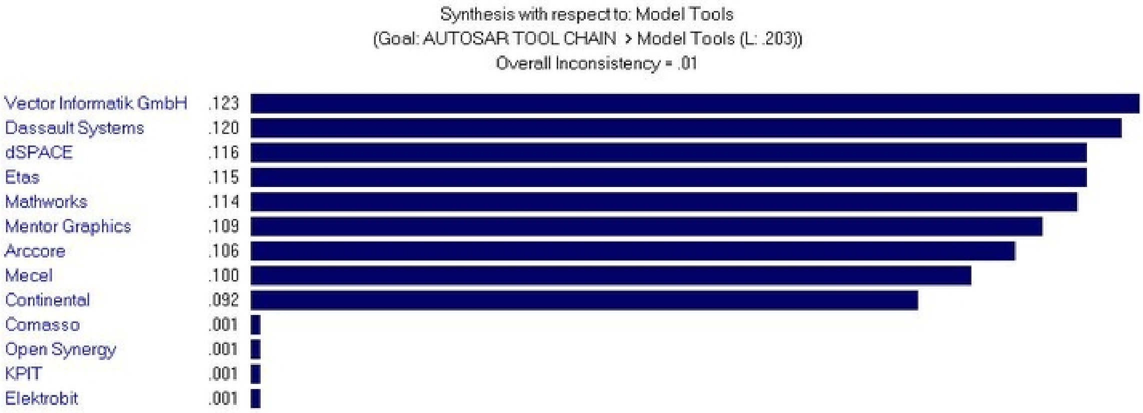


Figure A.5: Synthesis graph w.r.t. Model tools

ASW Tools

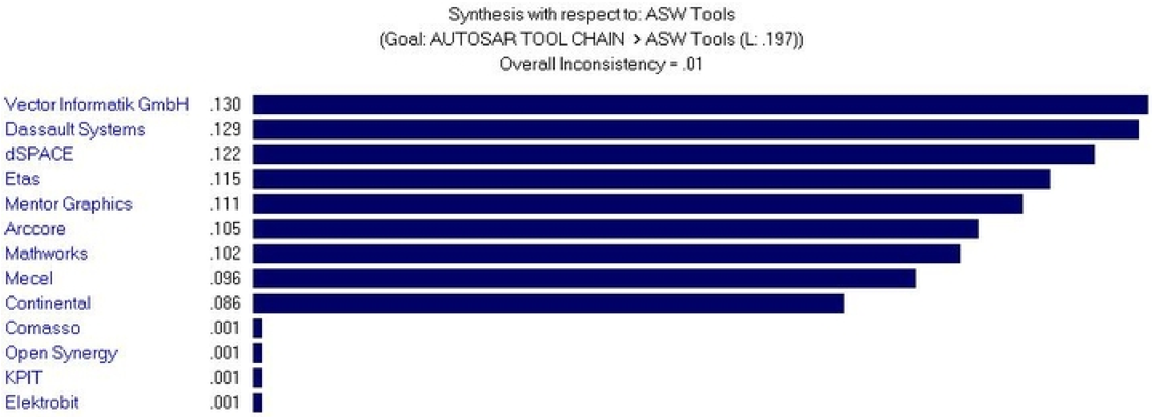


Figure A.6: Synthesis graph w.r.t. ASW tools

BSW Tools

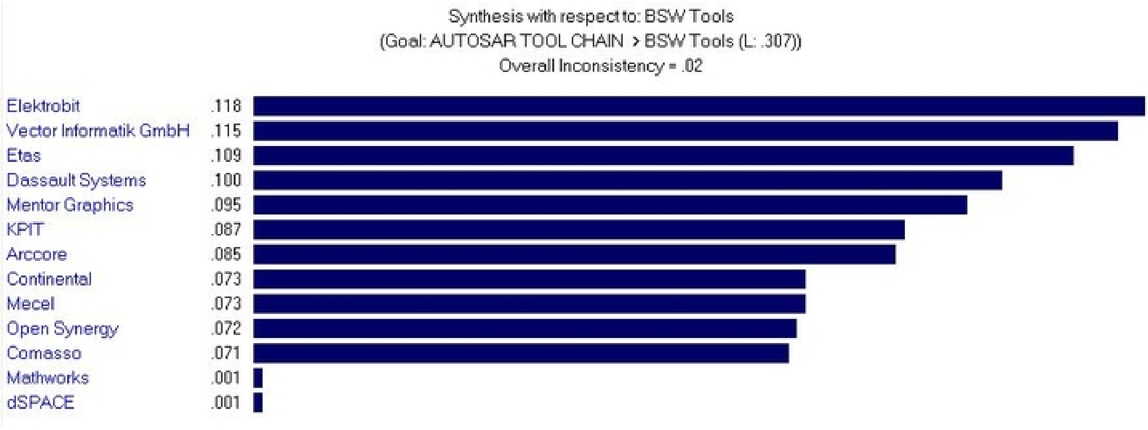


Figure A.7: Synthesis graph w.r.t. BSW tools

RTE Tools

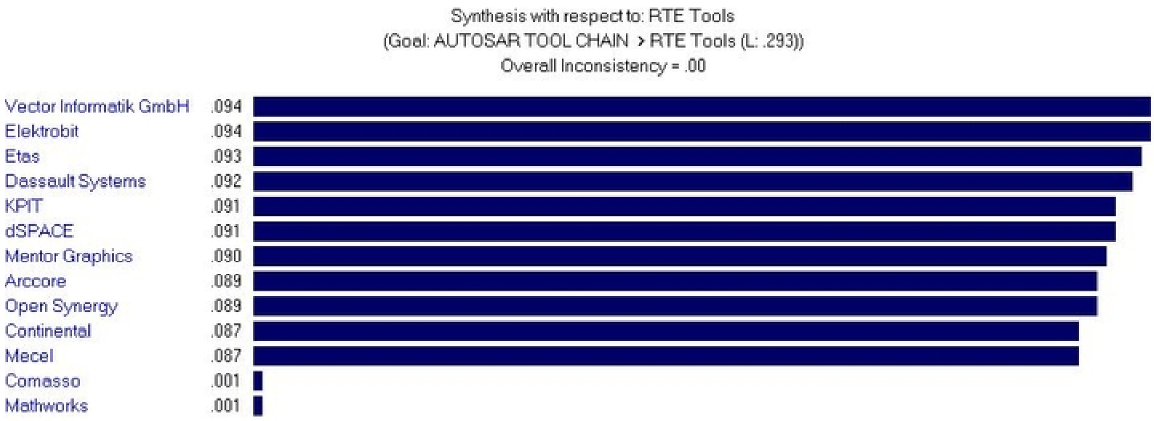
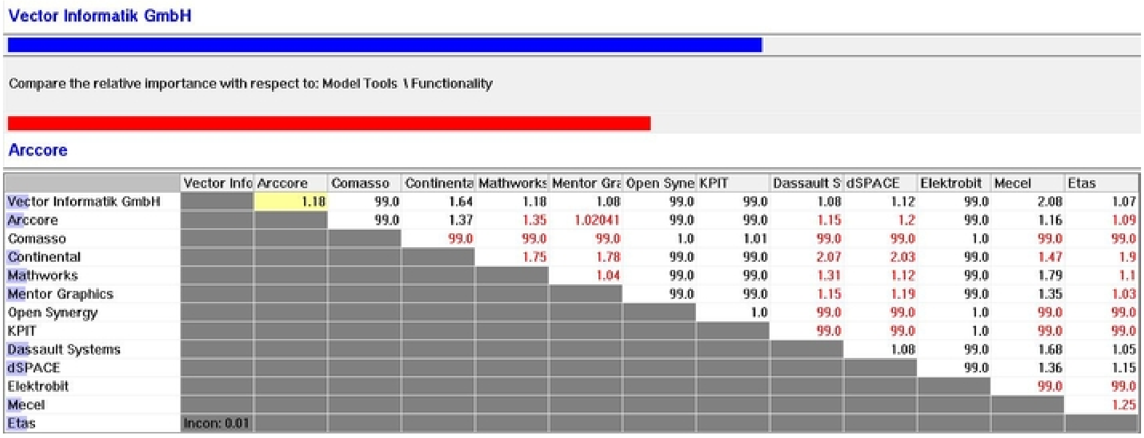


Figure A.8: Synthesis graph w.r.t. RTE tools

A.2.2 Overall synthesis results for each tool criteria Criteria - Functionality

The following synthesis graphs are w.r.t. specific criteria for each sub-goal (refer Figure 4.10). In Figure A.10, synthesis graph is w.r.t. the criteria - Functionality for Model Tools.

Model Tools



Vector Informatik GmbH

Compare the relative importance with respect to: Model Tools \ Functionality

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.18	99.0	1.64	1.18	1.08	99.0	99.0	1.08	1.12	99.0	2.08	1.07
Arccore			99.0	1.37	1.35	1.02041	99.0	99.0	1.15	1.2	99.0	1.16	1.09
Comasso				99.0	99.0	99.0	1.0	1.01	99.0	99.0	1.0	99.0	99.0
Continental					1.75	1.78	99.0	99.0	2.07	2.03	99.0	1.47	1.9
Mathworks						1.04	99.0	99.0	1.31	1.12	99.0	1.79	1.1
Mentor Graphics							99.0	99.0	1.15	1.19	99.0	1.35	1.03
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.08	99.0	1.68	1.05
dSPACE											99.0	1.36	1.15
Elektrobit												99.0	99.0
Mecel													1.25
Etas	Incon: 0.01												

Figure A.9: Pair-wise comparisons for the criteria Functionality of Model Tools

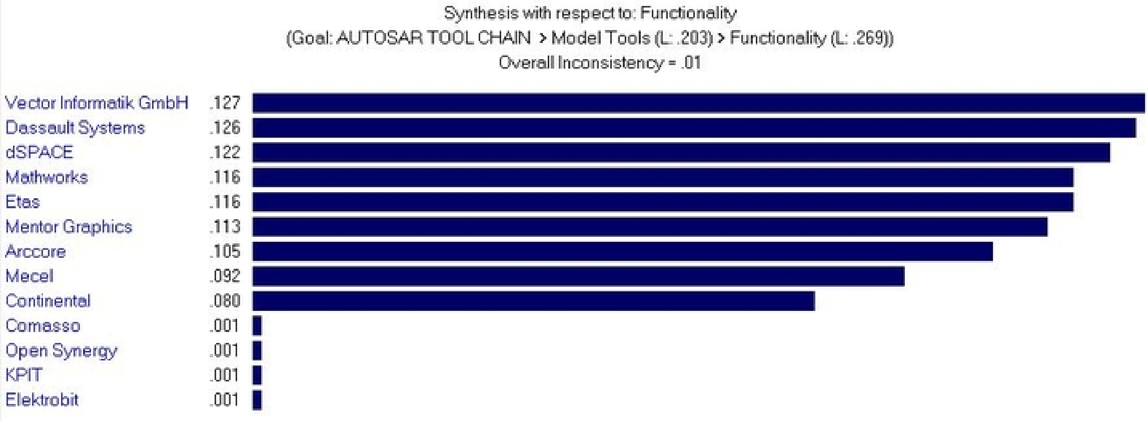
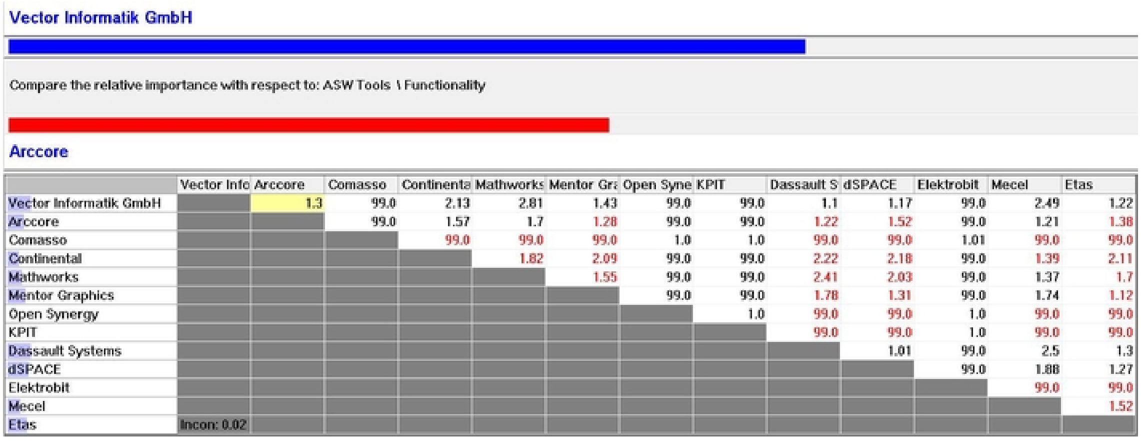


Figure A.10: Synthesis w.r.t. criteria - Functionality of Model Tools

Each tool alternative is compared with another tool alternative (e.g. Vector Informatik GmbH (blue band) is first compared with ArcCore (red band) w.r.t. functionality aspect for Model Tools). Vector's modeling tool (PREEvision) ranks higher than ArcCore's modeling tool, when compared to the features based on Functionality. The blue and red bands are adjusted accordingly. Vector when compared with COMASSO (which does not provide any modeling features), Vector's modeling tool gets the highest rank while COMASSO's modeling tool is given the least rank (i.e. the blue band is placed to the extreme right while the red band is placed to extreme left). Tools that ranked equally (e.g. both COMASSO and Open Synergy does not provide any modeling tool features) were given a value 1.0, i.e. the blue and red bands were positioned in the middle which signifies equal importance.

Accordingly, other tool alternatives were compared and assessed based on each tool criteria for each sub-goal.

ASW Tools



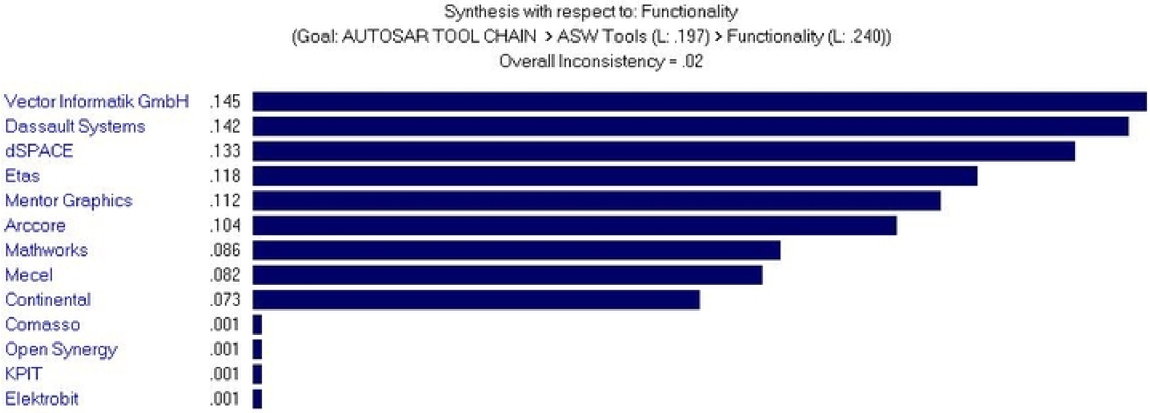
Vector Informatik GmbH

Compare the relative importance with respect to: ASW Tools \ Functionality

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.3	99.0	2.13	2.81	1.43	99.0	99.0	1.1	1.17	99.0	2.49	1.22
Arccore			99.0	1.57	1.7	1.28	99.0	99.0	1.22	1.52	99.0	1.21	1.38
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.01	99.0	99.0
Continental					1.82	2.09	99.0	99.0	2.22	2.18	99.0	1.39	2.11
Mathworks						1.55	99.0	99.0	2.41	2.03	99.0	1.37	1.7
Mentor Graphics							99.0	99.0	1.78	1.31	99.0	1.74	1.12
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.01	99.0	2.5	1.3
dSPACE											99.0	1.88	1.27
Elektrobit												99.0	99.0
Mecel													1.52
Etas	Incon: 0.02												

Figure A.11: Pair-wise comparisons for the criteria Functionality of ASW Tools



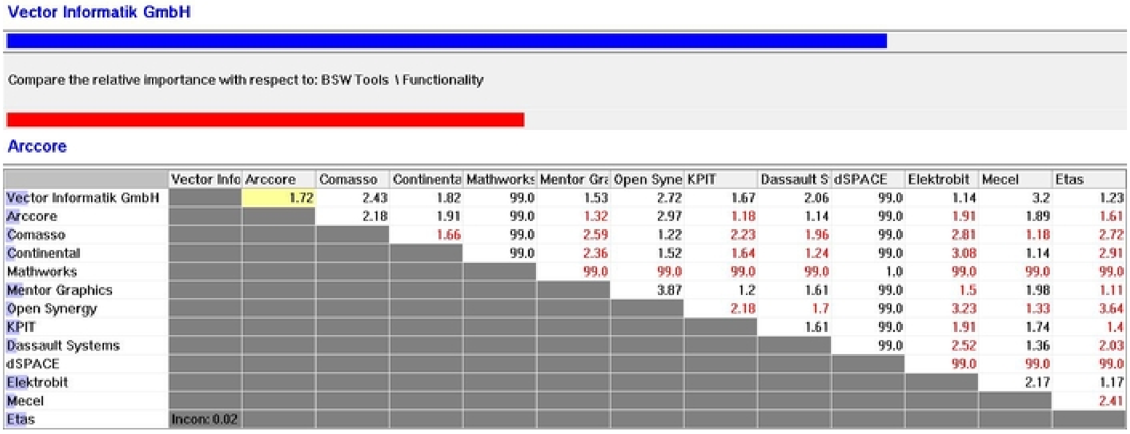
Synthesis with respect to: Functionality

(Goal: AUTOSAR TOOL CHAIN > ASW Tools (L: .197) > Functionality (L: .240))

Overall Inconsistency = .02

Figure A.12: Synthesis w.r.t. criteria - Functionality of ASW Tools

BSW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: BSW Tools \ Functionality

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.72	2.43	1.82	99.0	1.53	2.72	1.67	2.06	99.0	1.14	3.2	1.23
Arccore			2.18	1.91	99.0	1.32	2.97	1.18	1.14	99.0	1.91	1.89	1.61
Comasso				1.66	99.0	2.59	1.22	2.23	1.96	99.0	2.81	1.18	2.72
Continental					99.0	2.36	1.52	1.64	1.24	99.0	3.08	1.14	2.91
Mathworks						99.0	99.0	99.0	99.0	1.0	99.0	99.0	99.0
Mentor Graphics							3.87	1.2	1.61	99.0	1.5	1.98	1.11
Open Synergy								2.18	1.7	99.0	3.23	1.33	3.64
KPIT									1.61	99.0	1.91	1.74	1.4
Dassault Systems										99.0	2.52	1.36	2.03
dSPACE											99.0	99.0	99.0
Elektrobit												2.17	1.17
Mecel													2.41
Etas	Incon: 0.02												

Figure A.13: Pair-wise comparisons for the criteria Functionality of BSW Tools

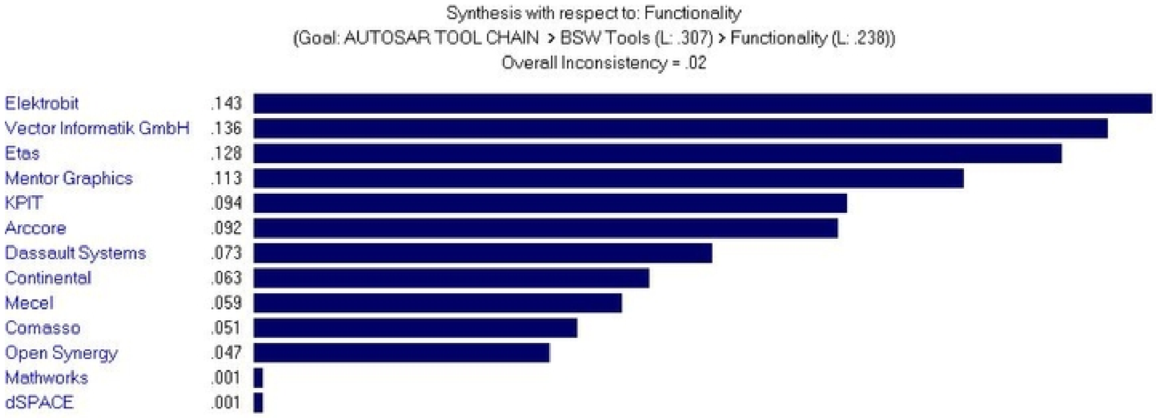
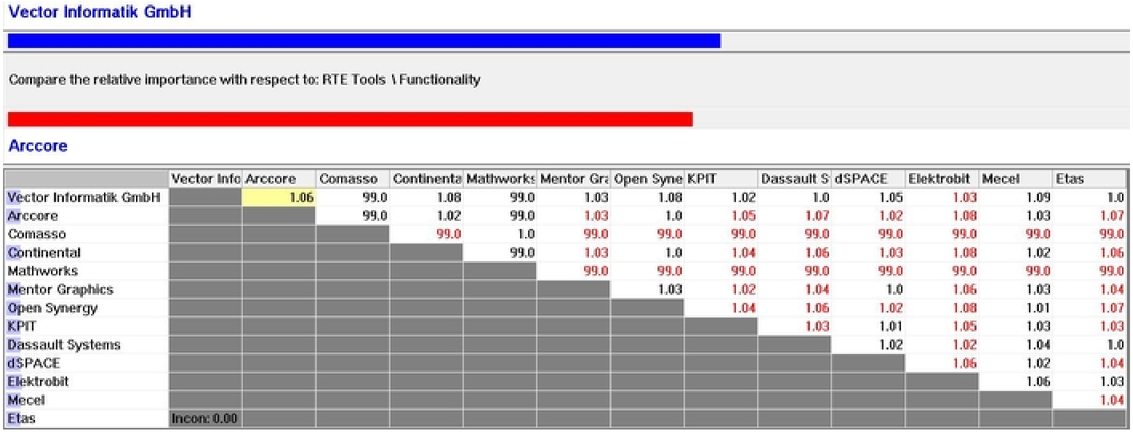


Figure A.14: Synthesis w.r.t. criteria - Functionality of BSW Tools

RTE Tools



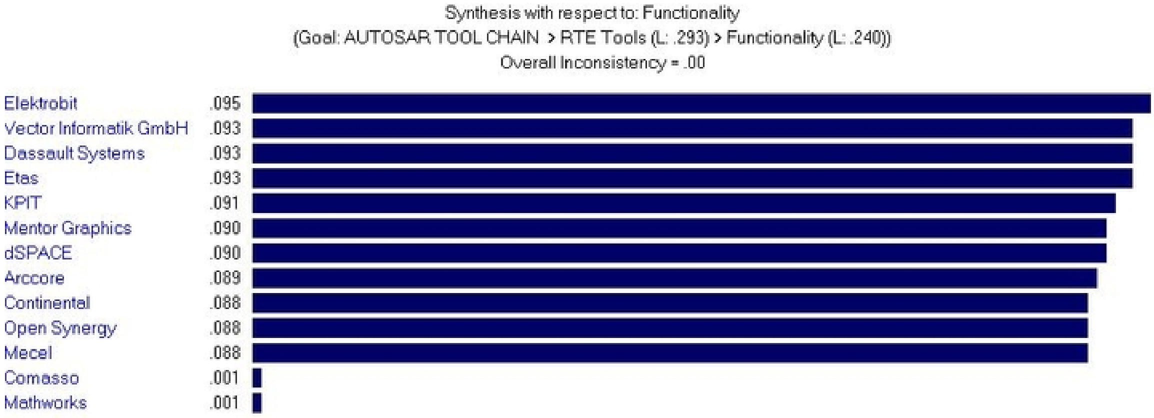
Vector Informatik GmbH

Compare the relative importance with respect to: RTE Tools \ Functionality

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.06	99.0	1.08	99.0	1.03	1.08	1.02	1.0	1.05	1.03	1.09	1.0
Arccore			99.0	1.02	99.0	1.03	1.0	1.05	1.07	1.02	1.08	1.03	1.07
Comasso				99.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Continental					99.0	1.03	1.0	1.04	1.06	1.03	1.08	1.02	1.06
Mathworks						99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Mentor Graphics							1.03	1.02	1.04	1.0	1.06	1.03	1.04
Open Synergy								1.04	1.06	1.02	1.08	1.01	1.07
KPIT									1.03	1.01	1.05	1.03	1.03
Dassault Systems										1.02	1.02	1.04	1.0
dSPACE											1.06	1.02	1.04
Elektrobit												1.06	1.03
Mecel													1.04
Etas	Incon: 0.00												

Figure A.15: Pair-wise comparisons for the criteria Functionality of RTE Tools

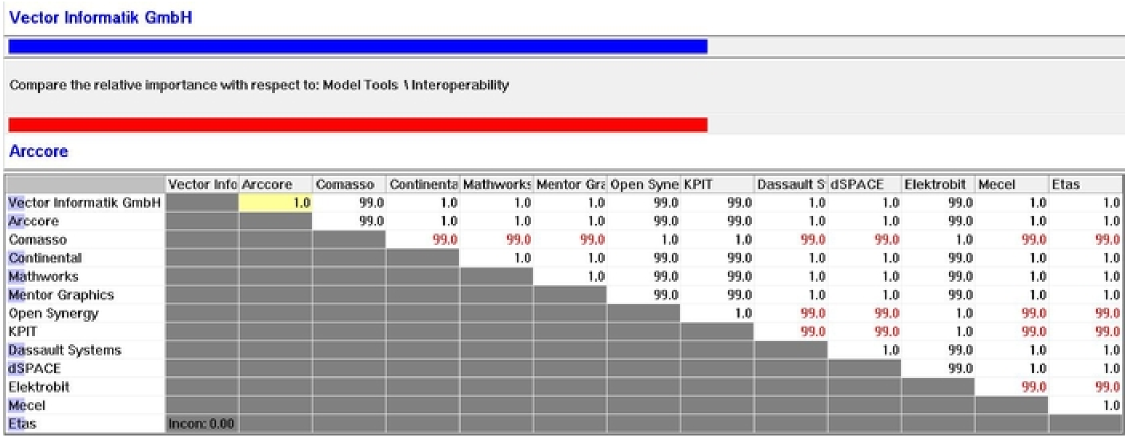


Synthesis with respect to: Functionality
(Goal: AUTOSAR TOOL CHAIN > RTE Tools (L: .293) > Functionality (L: .240))
Overall Inconsistency = .00

Figure A.16: Synthesis w.r.t. criteria - Functionality of RTE Tools

A.2.3 Criteria - Interoperability

Model Tools



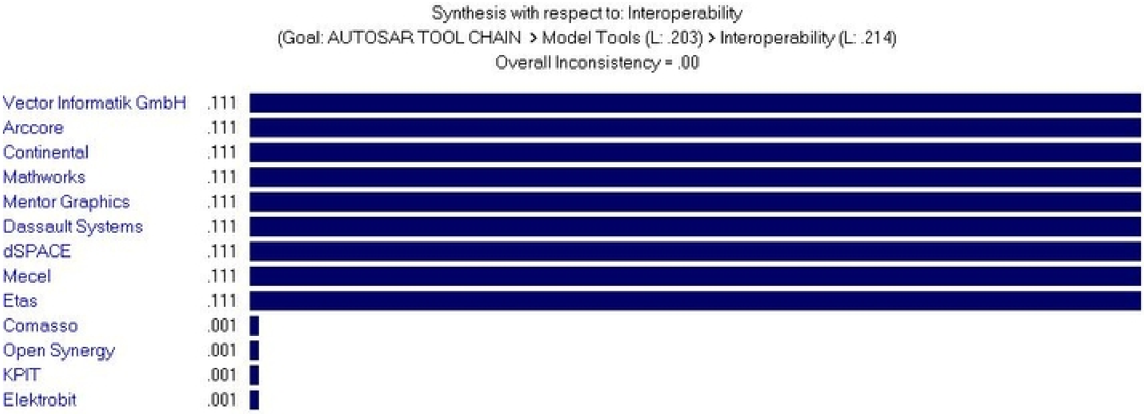
Vector Informatik GmbH

Compare the relative importance with respect to: Model Tools \ Interoperability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.0	99.0	1.0	1.0	1.0	99.0	99.0	1.0	1.0	99.0	1.0	1.0
Arccore			99.0	1.0	1.0	1.0	99.0	99.0	1.0	1.0	99.0	1.0	1.0
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.0	99.0	99.0
Continental					1.0	1.0	99.0	99.0	1.0	1.0	99.0	1.0	1.0
Mathworks						1.0	99.0	99.0	1.0	1.0	99.0	1.0	1.0
Mentor Graphics							99.0	99.0	1.0	1.0	99.0	1.0	1.0
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.0	99.0	1.0	1.0
dSPACE											99.0	1.0	1.0
Elektrobit												99.0	99.0
Mecel													1.0
Etas	Incon: 0.00												

Figure A.17: Pair-wise comparisons for the criteria Interoperability of Model Tools



Synthesis with respect to: Interoperability

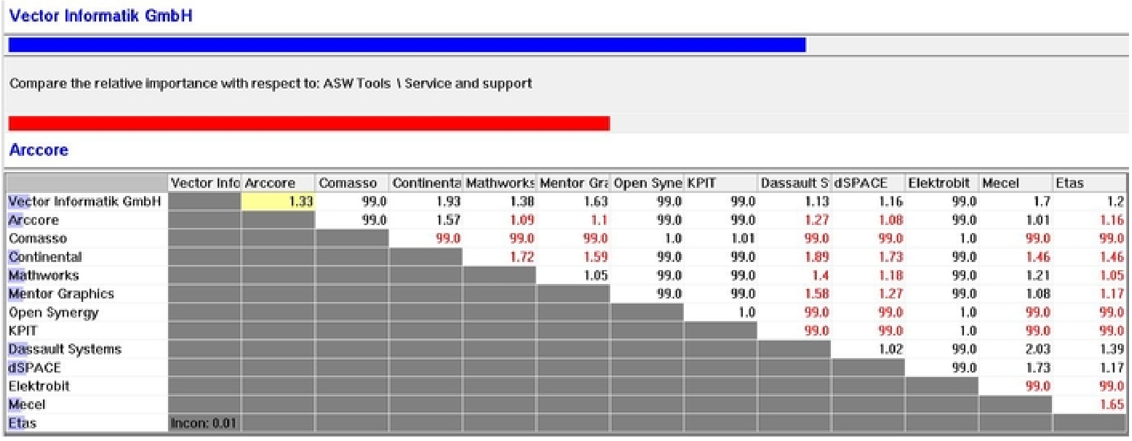
(Goal: AUTOSAR TOOL CHAIN > Model Tools (L: .203) > Interoperability (L: .214)

Overall Inconsistency = .00

Vector Informatik GmbH	.111
Arccore	.111
Continental	.111
Mathworks	.111
Mentor Graphics	.111
Dassault Systems	.111
dSPACE	.111
Mecel	.111
Etas	.111
Comasso	.001
Open Synergy	.001
KPIT	.001
Elektrobit	.001

Figure A.18: Synthesis w.r.t. criteria - Interoperability of Model Tools

ASW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: ASW Tools \ Service and support

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.33	99.0	1.93	1.38	1.63	99.0	99.0	1.13	1.16	99.0	1.7	1.2
Arccore			99.0	1.57	1.09	1.1	99.0	99.0	1.27	1.08	99.0	1.01	1.16
Comasso				99.0	99.0	99.0	1.0	1.01	99.0	99.0	1.0	99.0	99.0
Continental					1.72	1.59	99.0	99.0	1.89	1.73	99.0	1.46	1.46
Mathworks						1.05	99.0	99.0	1.4	1.18	99.0	1.21	1.05
Mentor Graphics							99.0	99.0	1.58	1.27	99.0	1.08	1.17
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.02	99.0	2.03	1.39
dSPACE											99.0	1.73	1.17
Elektrobit												99.0	99.0
Mecel													1.65
Etas	Incon: 0.01												

Figure A.19: Pair-wise comparisons for the criteria Interoperability of ASW Tools

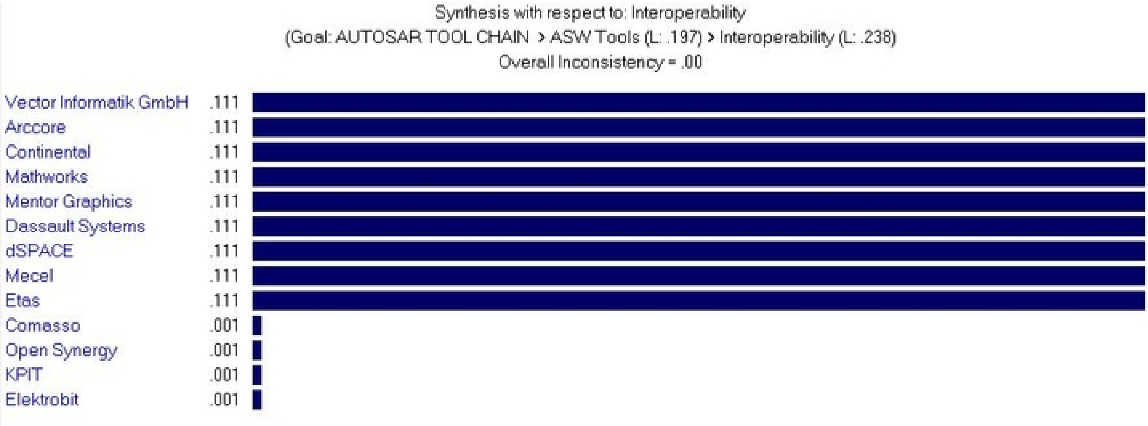
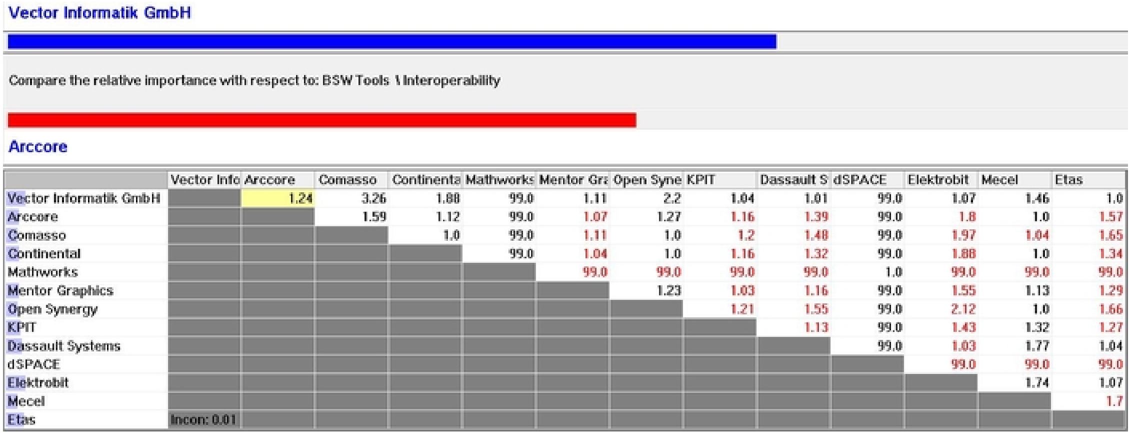


Figure A.20: Synthesis w.r.t. criteria - Interoperability of ASW Tools

BSW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: BSW Tools \ Interoperability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.24	3.26	1.88	99.0	1.11	2.2	1.04	1.01	99.0	1.07	1.46	1.0
Arccore			1.59	1.12	99.0	1.07	1.27	1.16	1.39	99.0	1.8	1.0	1.57
Comasso				1.0	99.0	1.11	1.0	1.2	1.48	99.0	1.97	1.04	1.65
Continental					99.0	1.04	1.0	1.16	1.32	99.0	1.88	1.0	1.34
Mathworks						99.0	99.0	99.0	99.0	1.0	99.0	99.0	99.0
Mentor Graphics							1.23	1.03	1.16	99.0	1.55	1.13	1.29
Open Synergy								1.21	1.55	99.0	2.12	1.0	1.66
KPIT									1.13	99.0	1.43	1.32	1.27
Dassault Systems										99.0	1.03	1.77	1.04
dSPACE											99.0	99.0	99.0
Elektrobit												1.74	1.07
Mecel													1.7
Etas	Incon: 0.01												

Figure A.21: Pair-wise comparisons for the criteria Interoperability of BSW Tools

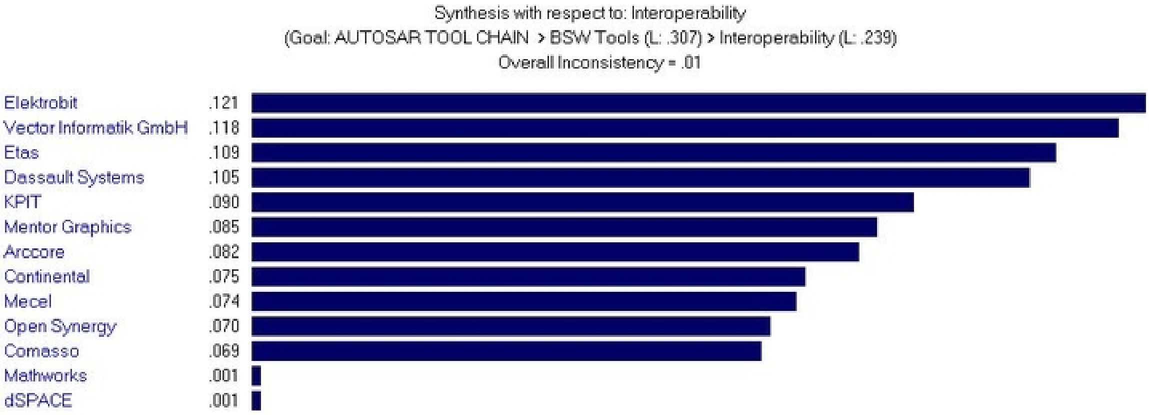
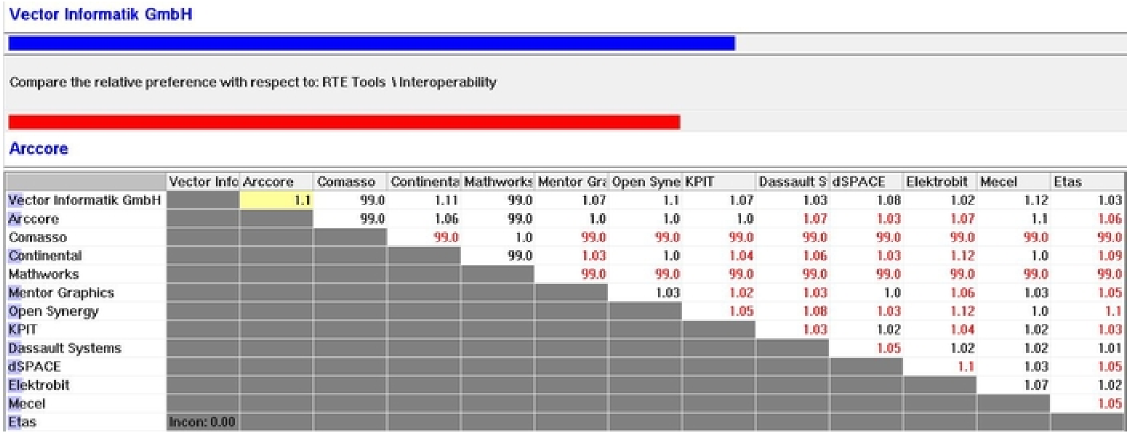


Figure A.22: Synthesis w.r.t. criteria - Interoperability of BSW Tools

RTE Tools



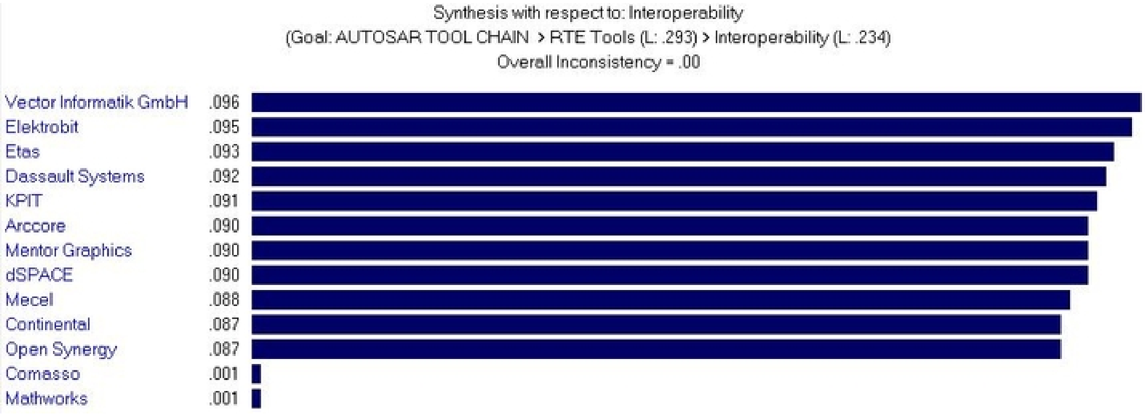
Vector Informatik GmbH

Compare the relative preference with respect to: RTE Tools \ Interoperability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.1	99.0	1.11	99.0	1.07	1.1	1.07	1.03	1.08	1.02	1.12	1.03
Arccore			99.0	1.06	99.0	1.0	1.0	1.0	1.07	1.03	1.07	1.1	1.06
Comasso				99.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Continental					99.0	1.03	1.0	1.04	1.06	1.03	1.12	1.0	1.09
Mathworks						99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Mentor Graphics							1.03	1.02	1.03	1.0	1.06	1.03	1.05
Open Synergy								1.05	1.08	1.03	1.12	1.0	1.1
KPIT									1.03	1.02	1.04	1.02	1.03
Dassault Systems										1.05	1.02	1.02	1.01
dSPACE											1.1	1.03	1.05
Elektrobit												1.07	1.02
Mecel													1.05
Etas	Incon: 0.00												

Figure A.23: Pair-wise comparisons for the criteria Interoperability of RTE Tools



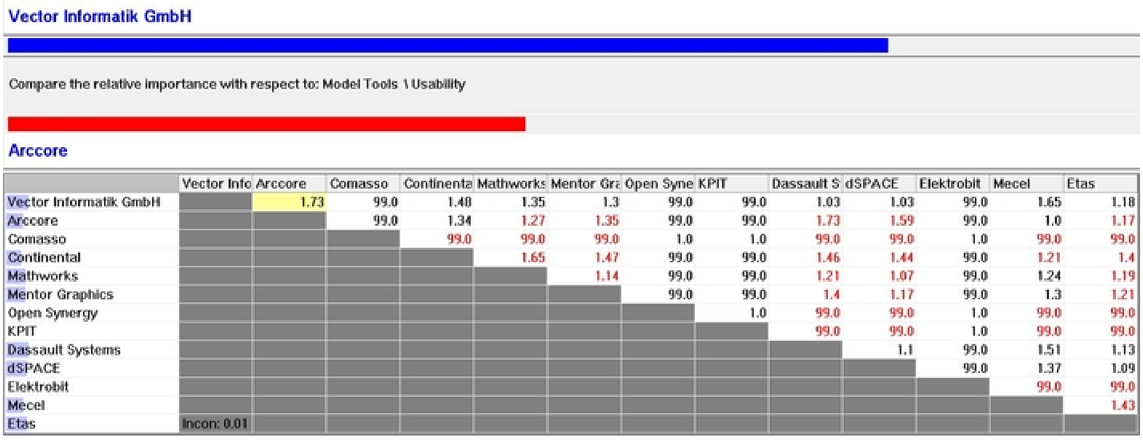
Synthesis with respect to: Interoperability
(Goal: AUTOSAR TOOL CHAIN > RTE Tools (L: .293) > Interoperability (L: .234)
Overall Inconsistency = .00

Vector Informatik GmbH	.096
Elektrobit	.095
Etas	.093
Dassault Systems	.092
KPIT	.091
Arccore	.090
Mentor Graphics	.090
dSPACE	.090
Mecel	.088
Continental	.087
Open Synergy	.087
Comasso	.001
Mathworks	.001

Figure A.24: Synthesis w.r.t. criteria - Interoperability of RTE Tools

A.2.4 Criteria - Usability

Model Tools



Compare the relative importance with respect to: Model Tools \ Usability

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.73	99.0	1.48	1.35	1.3	99.0	99.0	1.03	1.03	99.0	1.65	1.18
Arccore			99.0	1.34	1.27	1.35	99.0	99.0	1.73	1.59	99.0	1.0	1.17
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.0	99.0	99.0
Continental					1.65	1.47	99.0	99.0	1.46	1.44	99.0	1.21	1.4
Mathworks						1.14	99.0	99.0	1.21	1.07	99.0	1.24	1.19
Mentor Graphics							99.0	99.0	1.4	1.17	99.0	1.3	1.21
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.1	99.0	1.51	1.13
dSPACE											99.0	1.37	1.09
Elektrobit												99.0	99.0
Mecel													1.43
Etas	Incon: 0.01												

Figure A.25: Pair-wise comparisons for the criteria Usability of Model Tools

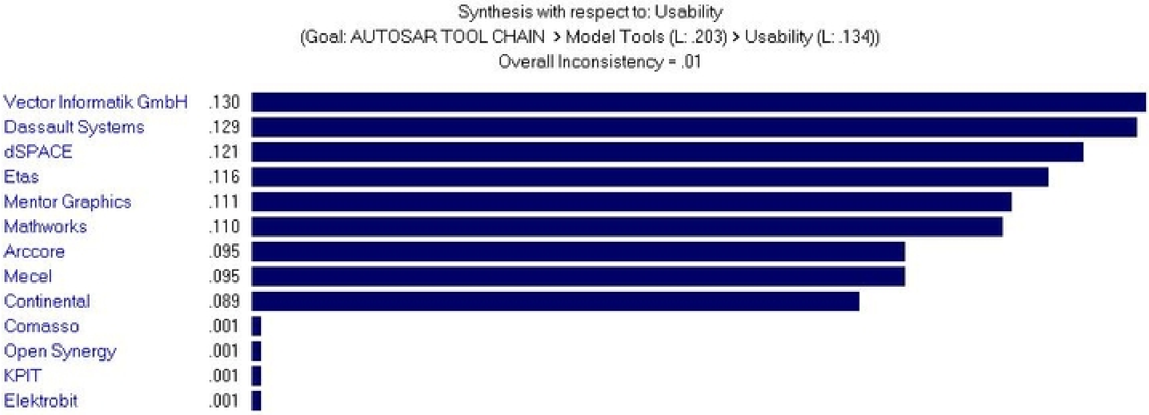
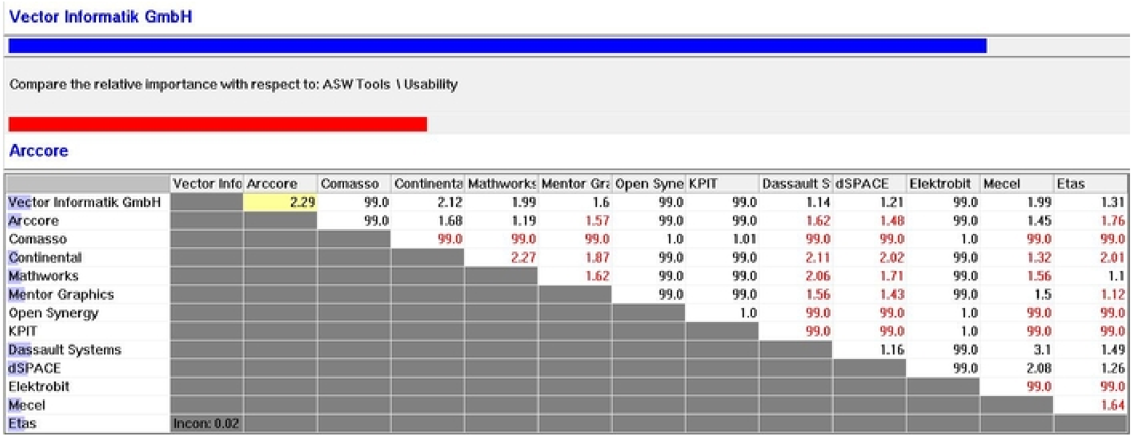


Figure A.26: Synthesis w.r.t. criteria - Usability of Model Tools

ASW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: ASW Tools \ Usability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		2.29	99.0	2.12	1.99	1.6	99.0	99.0	1.14	1.21	99.0	1.99	1.31
Arccore			99.0	1.68	1.19	1.57	99.0	99.0	1.62	1.48	99.0	1.45	1.76
Comasso				99.0	99.0	99.0	1.0	1.01	99.0	99.0	1.0	99.0	99.0
Continental					2.27	1.87	99.0	99.0	2.11	2.02	99.0	1.32	2.01
Mathworks						1.62	99.0	99.0	2.06	1.71	99.0	1.56	1.1
Mentor Graphics							99.0	99.0	1.56	1.43	99.0	1.5	1.12
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.16	99.0	3.1	1.49
dSPACE											99.0	2.08	1.26
Elektrobit												99.0	99.0
Mecel													1.64
Etas	Incon: 0.02												

Figure A.27: Pair-wise comparisons for the criteria Usability of ASW Tools

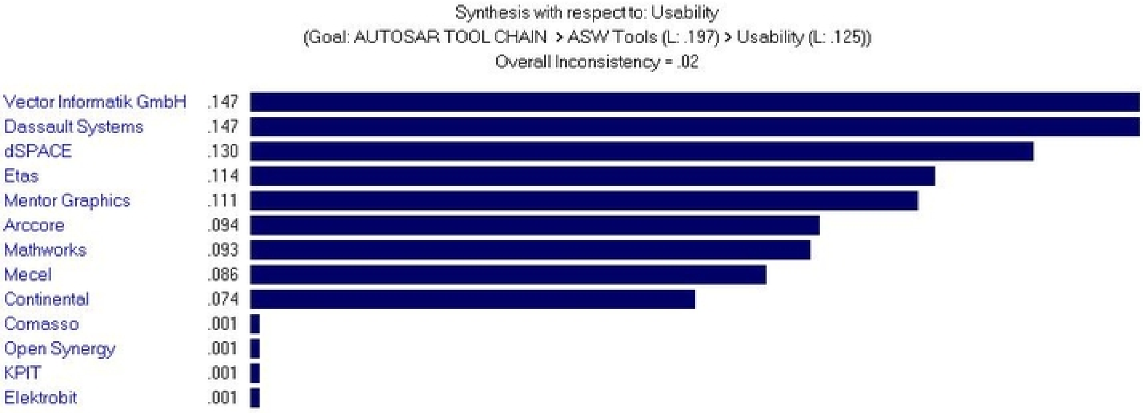
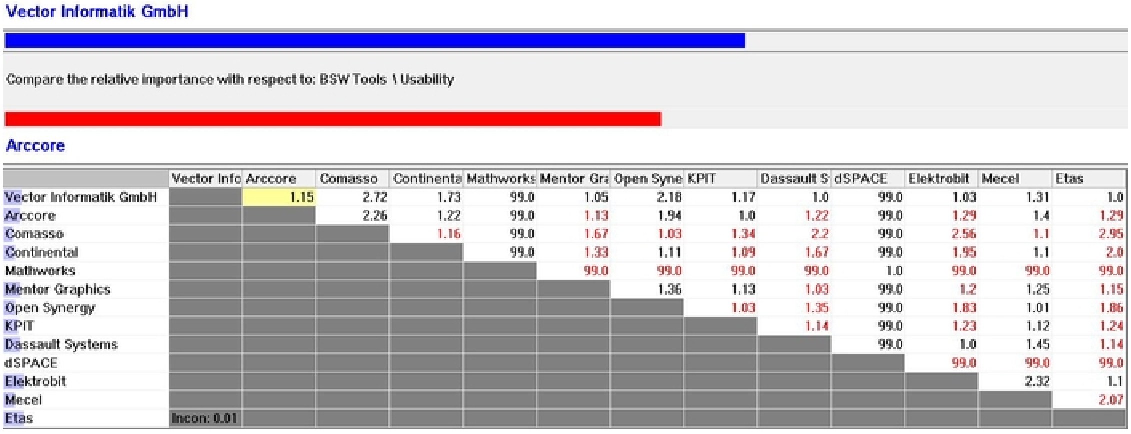


Figure A.28: Synthesis w.r.t. criteria - Usability of ASW Tools

BSW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: BSW Tools \ Usability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.15	2.72	1.73	99.0	1.05	2.18	1.17	1.0	99.0	1.03	1.31	1.0
Arccore			2.26	1.22	99.0	1.13	1.94	1.0	1.22	99.0	1.29	1.4	1.29
Comasso				1.16	99.0	1.67	1.03	1.34	2.2	99.0	2.56	1.1	2.95
Continental					99.0	1.33	1.11	1.09	1.67	99.0	1.95	1.1	2.0
Mathworks						99.0	99.0	99.0	99.0	1.0	99.0	99.0	99.0
Mentor Graphics							1.36	1.13	1.03	99.0	1.2	1.25	1.15
Open Synergy								1.03	1.35	99.0	1.83	1.01	1.86
KPIT									1.14	99.0	1.23	1.12	1.24
Dassault Systems										99.0	1.0	1.45	1.14
dSPACE											99.0	99.0	99.0
Elektrobit												2.32	1.1
Mecel													2.07
Etas	Incon: 0.01												

Figure A.29: Pair-wise comparisons for the criteria Usability of BSW Tools

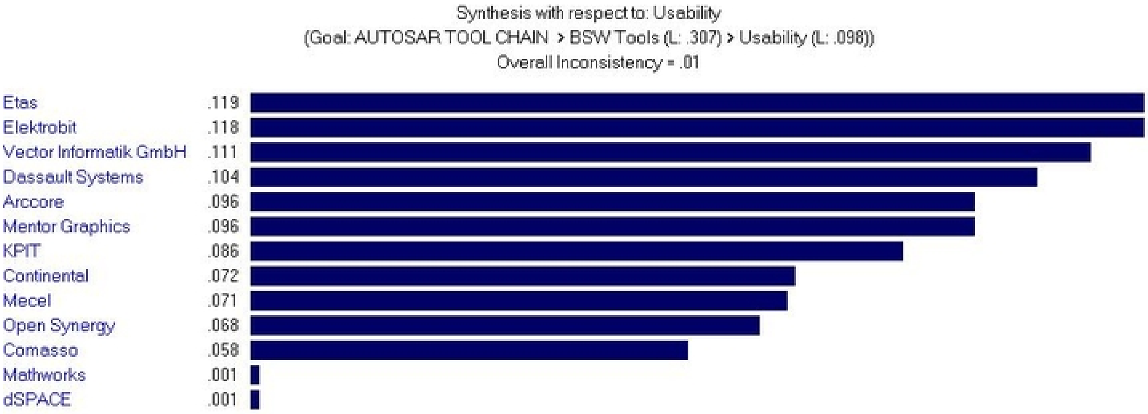
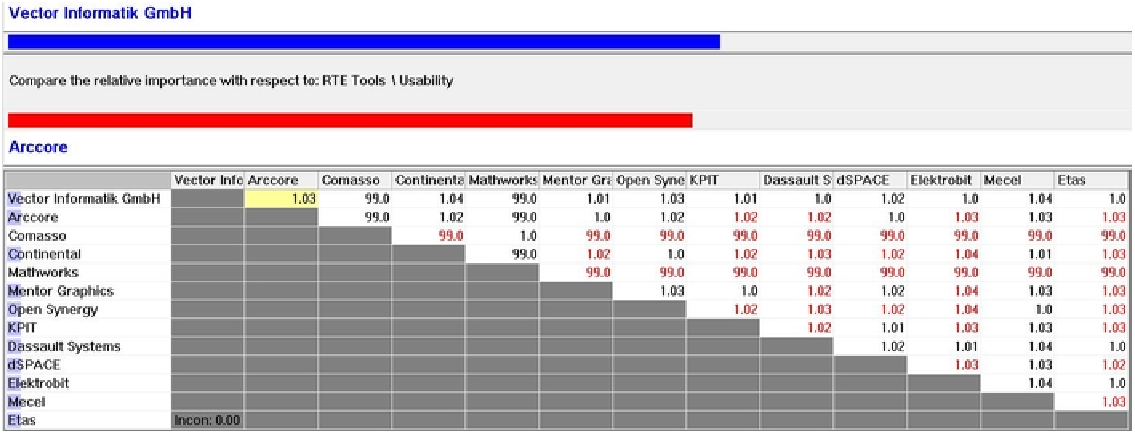


Figure A.30: Synthesis w.r.t. criteria - Usability of BSW Tools

RTE Tools



Vector Informatik GmbH

Compare the relative importance with respect to: RTE Tools \ Usability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.03	99.0	1.04	99.0	1.01	1.03	1.01	1.0	1.02	1.0	1.04	1.0
Arccore			99.0	1.02	99.0	1.0	1.02	1.02	1.02	1.0	1.03	1.03	1.03
Comasso				99.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Continental					99.0	1.02	1.0	1.02	1.03	1.02	1.04	1.01	1.03
Mathworks						99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Mentor Graphics							1.03	1.0	1.02	1.02	1.04	1.03	1.03
Open Synergy								1.02	1.03	1.02	1.04	1.0	1.03
KPIT									1.02	1.01	1.03	1.03	1.03
Dassault Systems										1.02	1.01	1.04	1.0
dSPACE											1.03	1.03	1.02
Elektrobit												1.04	1.0
Mecel													1.03
Etas	Incon: 0.00												

Figure A.31: Pair-wise comparisons for the criteria Usability of RTE Tools

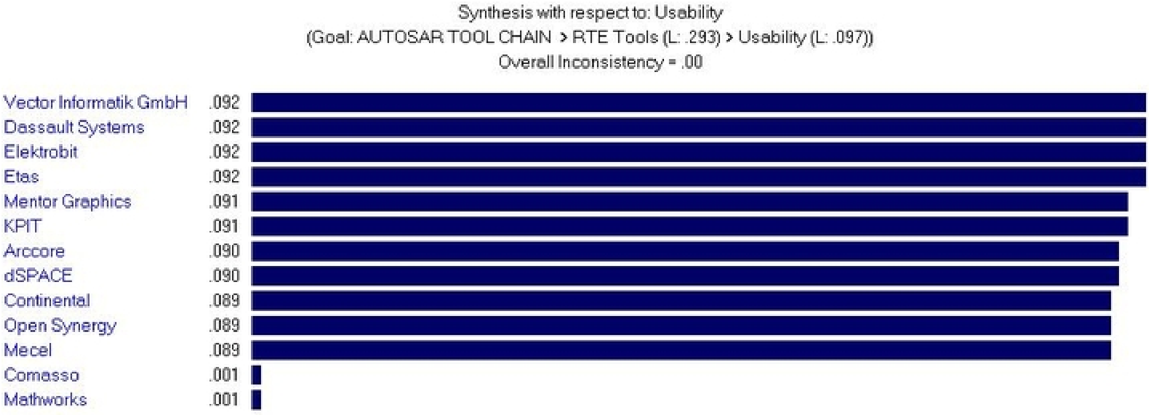
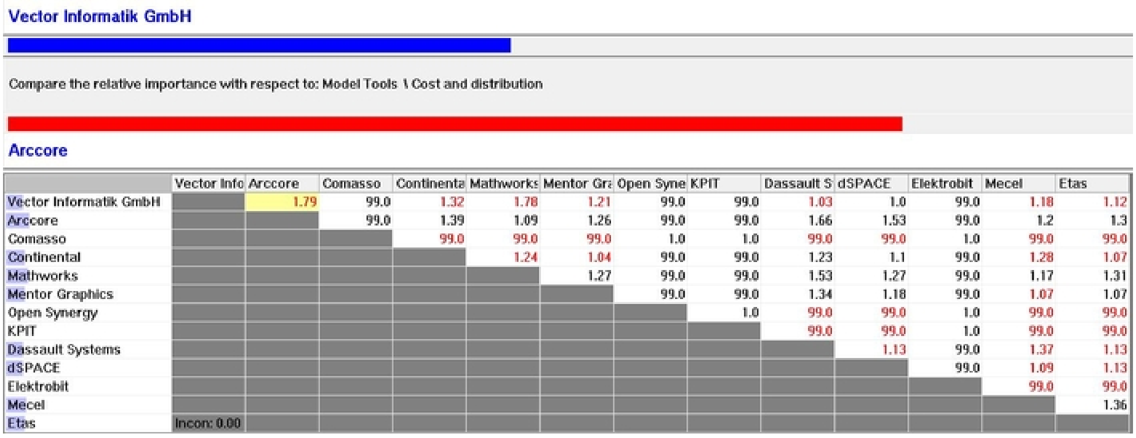


Figure A.32: Synthesis w.r.t. criteria - Usability of RTE Tools

A.2.5 Criteria - Cost and Distribution

Model Tools



Vector Informatik GmbH

Compare the relative importance with respect to: Model Tools \ Cost and distribution

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.79	99.0	1.32	1.78	1.21	99.0	99.0	1.03	1.0	99.0	1.18	1.12
Arccore			99.0	1.39	1.09	1.26	99.0	99.0	1.66	1.53	99.0	1.2	1.3
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.0	99.0	99.0
Continental					1.24	1.04	99.0	99.0	1.23	1.1	99.0	1.28	1.07
Mathworks						1.27	99.0	99.0	1.53	1.27	99.0	1.17	1.31
Mentor Graphics							99.0	99.0	1.34	1.18	99.0	1.07	1.07
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.13	99.0	1.37	1.13
dSPACE											99.0	1.09	1.13
Elektrobit												99.0	99.0
Mecel													1.36
Etas	Incon: 0.00												

Figure A.33: Pair-wise comparisons for the criteria Cost and Distribution of Model Tools

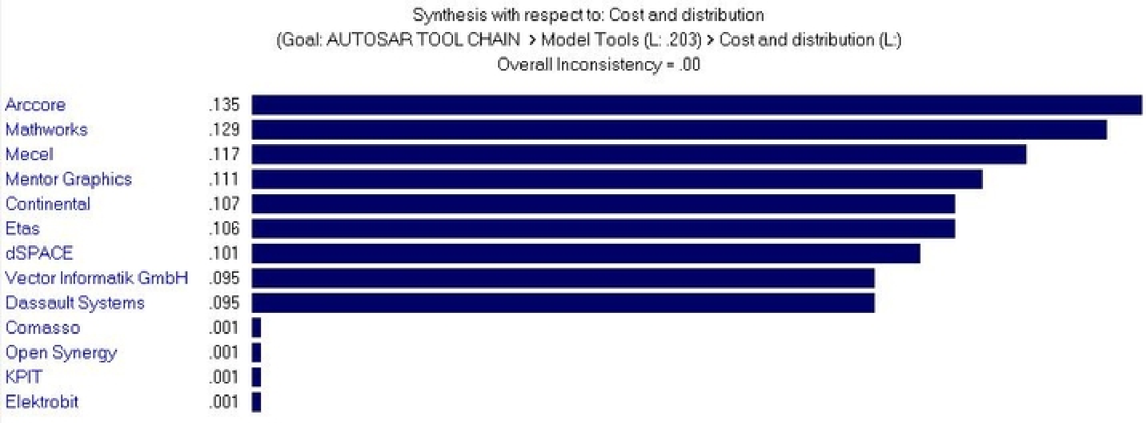
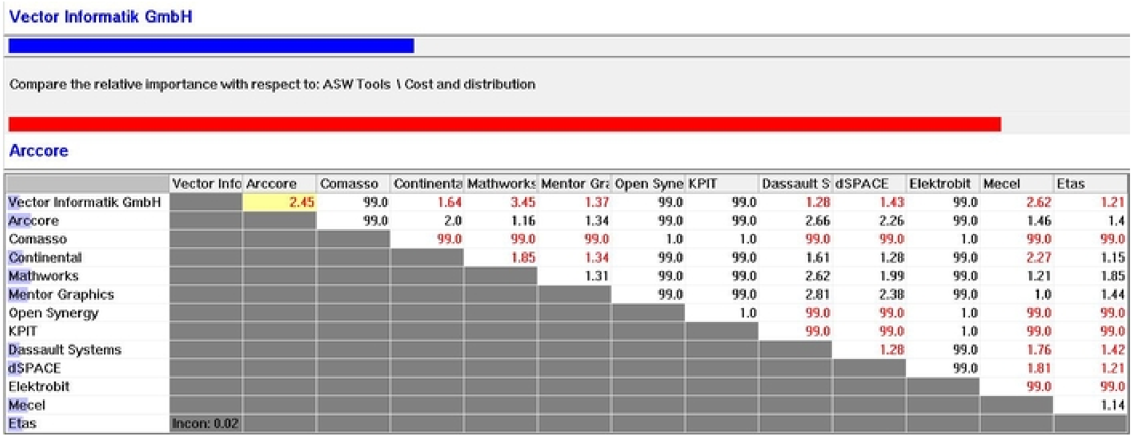


Figure A.34: Synthesis w.r.t. criteria - Cost and Distribution of Model Tools

ASW Tools



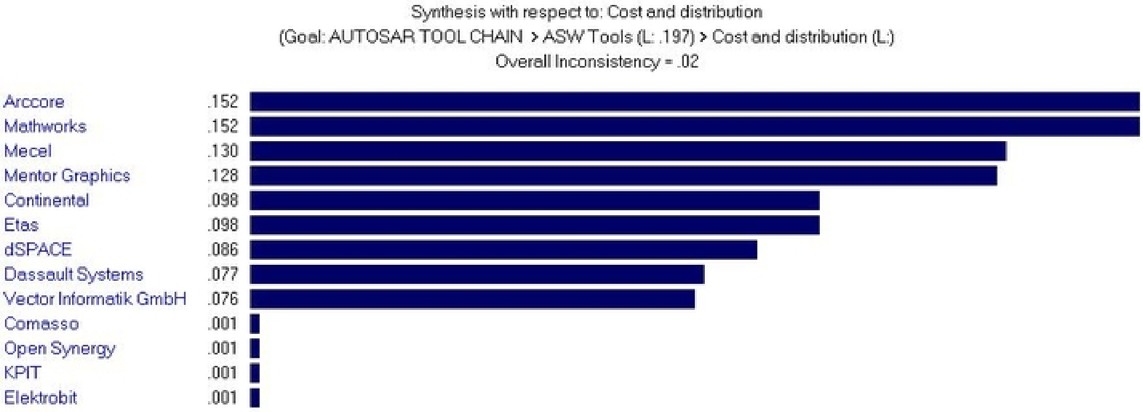
Vector Informatik GmbH

Compare the relative importance with respect to: ASW Tools \ Cost and distribution

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		2.45	99.0	1.64	3.45	1.37	99.0	99.0	1.28	1.43	99.0	2.62	1.21
Arccore			99.0	2.0	1.16	1.34	99.0	99.0	2.66	2.26	99.0	1.46	1.4
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.0	99.0	99.0
Continental					1.85	1.34	99.0	99.0	1.61	1.28	99.0	2.27	1.15
Mathworks						1.31	99.0	99.0	2.62	1.99	99.0	1.21	1.85
Mentor Graphics							99.0	99.0	2.81	2.38	99.0	1.0	1.44
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.28	99.0	1.76	1.42
dSPACE											99.0	1.81	1.21
Elektrobit												99.0	99.0
Mecel													1.14
Etas	Incon: 0.02												

Figure A.35: Pair-wise comparisons for the criteria Cost and Distribution of ASW Tools



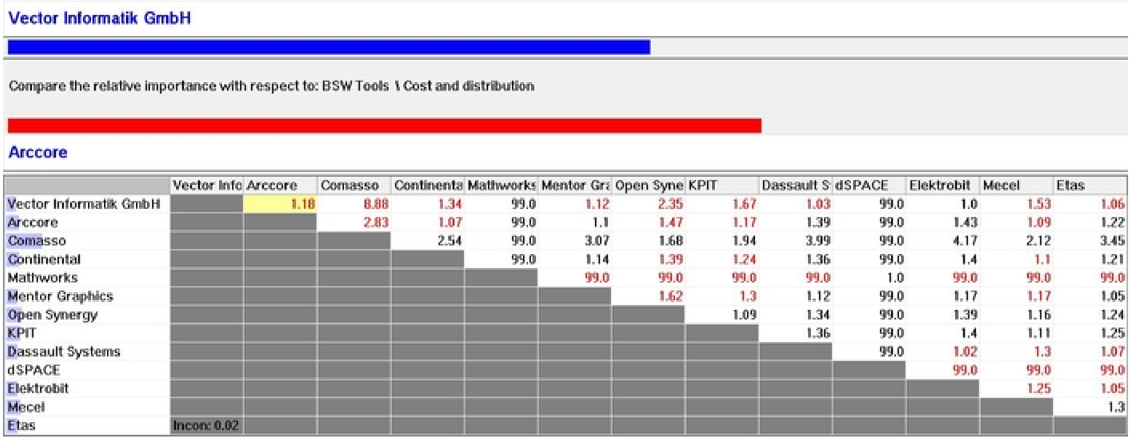
Synthesis with respect to: Cost and distribution

(Goal: AUTOSAR TOOL CHAIN > ASW Tools (L: .197) > Cost and distribution (L:)

Overall Inconsistency = .02

Figure A.36: Synthesis w.r.t. criteria - Cost and Distribution of ASW Tools

BSW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: BSW Tools \ Cost and distribution

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.18	8.88	1.34	99.0	1.12	2.35	1.67	1.03	99.0	1.0	1.53	1.06
Arccore			2.83	1.07	99.0	1.1	1.47	1.17	1.39	99.0	1.43	1.09	1.22
Comasso				2.54	99.0	3.07	1.68	1.94	3.99	99.0	4.17	2.12	3.45
Continental					99.0	1.14	1.39	1.24	1.36	99.0	1.4	1.1	1.21
Mathworks						99.0	99.0	99.0	99.0	1.0	99.0	99.0	99.0
Mentor Graphics							1.62	1.3	1.12	99.0	1.17	1.17	1.05
Open Synergy								1.09	1.34	99.0	1.39	1.16	1.24
KPIT									1.36	99.0	1.4	1.11	1.25
Dassault Systems										99.0	1.02	1.3	1.07
dSPACE											99.0	99.0	99.0
Elektrobit												1.25	1.05
Mecel													1.3
Etas	Incon: 0.02												

Figure A.37: Pair-wise comparisons for the criteria Cost and Distribution of BSW Tools

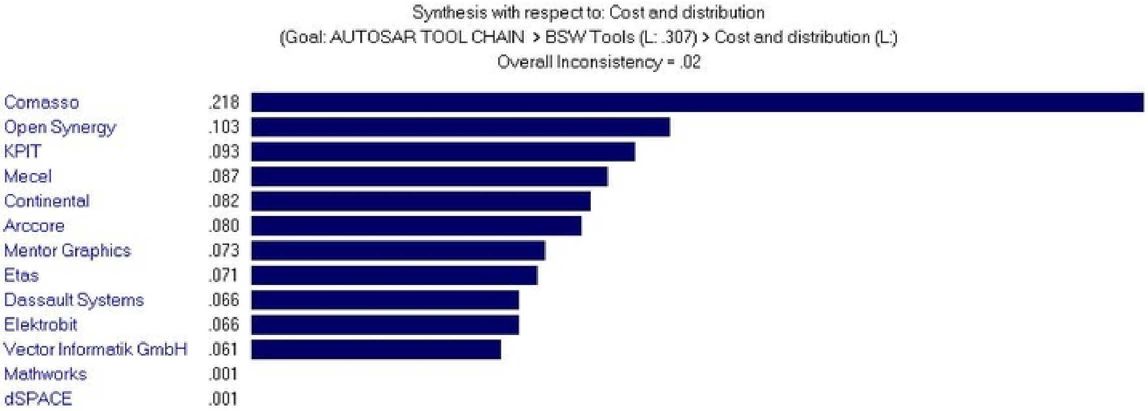
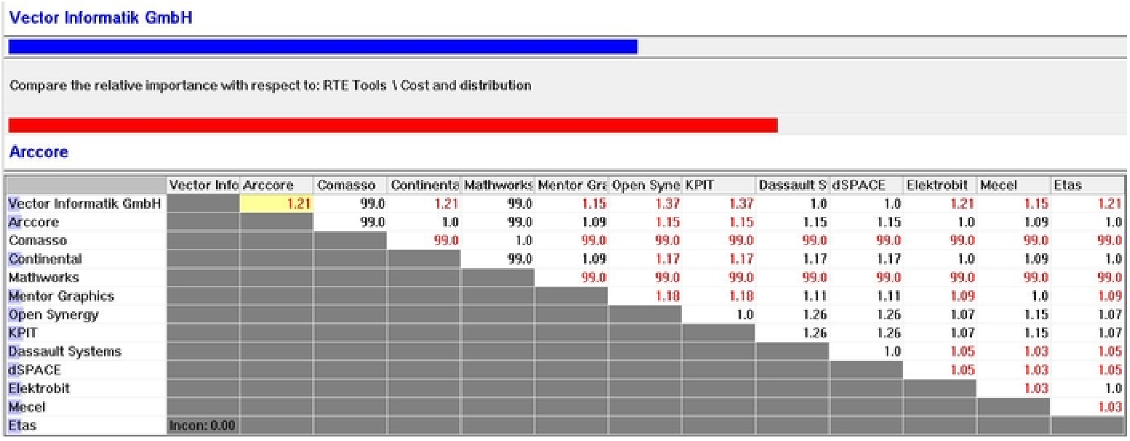


Figure A.38: Synthesis w.r.t. criteria - Cost and Distribution of BSW Tools

RTE Tools



Vector Informatik GmbH

Compare the relative importance with respect to: RTE Tools \ Cost and distribution

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.21	99.0	1.21	99.0	1.15	1.37	1.37	1.0	1.0	1.21	1.15	1.21
Arccore			99.0	1.0	99.0	1.09	1.15	1.15	1.15	1.15	1.0	1.09	1.0
Comasso				99.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Continental					99.0	1.09	1.17	1.17	1.17	1.17	1.0	1.09	1.0
Mathworks						99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Mentor Graphics							1.18	1.18	1.11	1.11	1.09	1.0	1.09
Open Synergy								1.0	1.26	1.26	1.07	1.15	1.07
KPIT									1.26	1.26	1.07	1.15	1.07
Dassault Systems										1.0	1.05	1.03	1.05
dSPACE											1.05	1.03	1.05
Elektrobit												1.03	1.0
Mecel													1.03
Etas	Incon: 0.00												

Figure A.39: Pair-wise comparisons for the criteria Cost and Distribution of RTE Tools

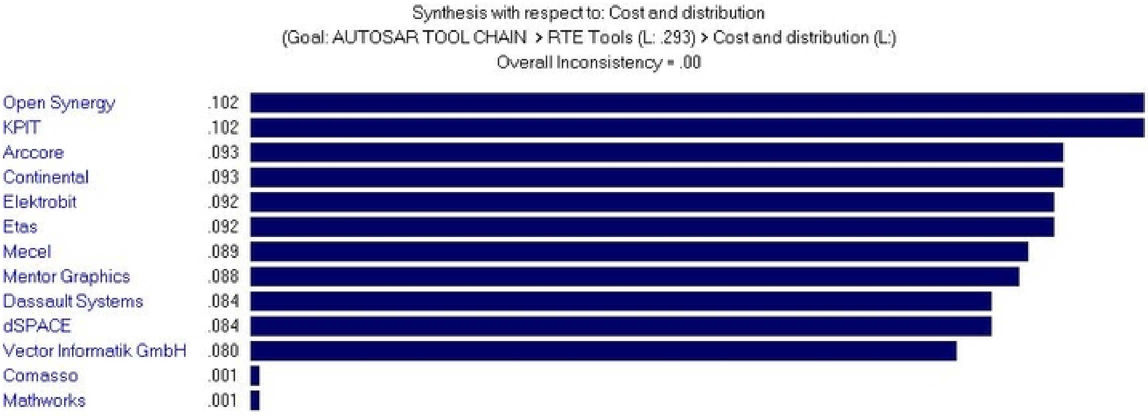
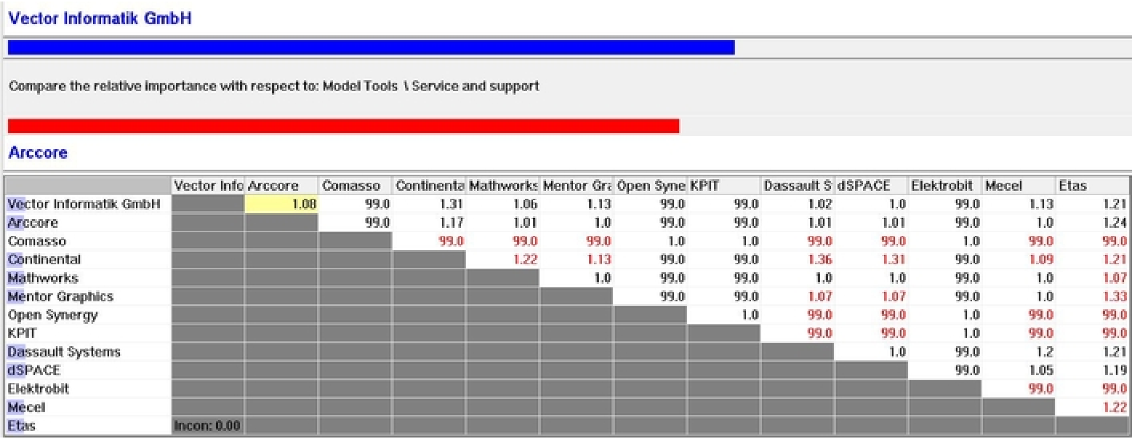


Figure A.40: Synthesis w.r.t. criteria - Cost and Distribution of RTE Tools

A.2.6 Criteria - Service and Support

Model Tools



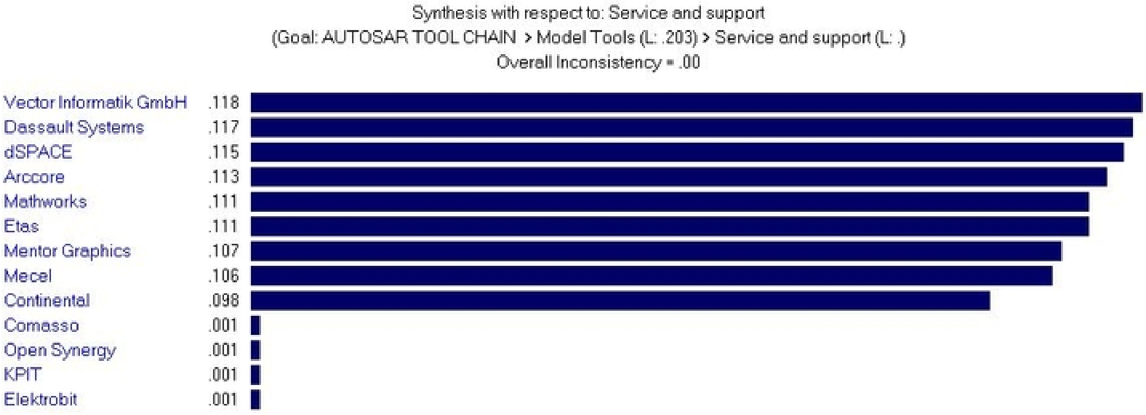
Vector Informatik GmbH

Compare the relative importance with respect to: Model Tools \ Service and support

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.08	99.0	1.31	1.06	1.13	99.0	99.0	1.02	1.0	99.0	1.13	1.21
Arccore			99.0	1.17	1.01	1.0	99.0	99.0	1.01	1.01	99.0	1.0	1.24
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.0	99.0	99.0
Continental					1.22	1.13	99.0	99.0	1.36	1.31	99.0	1.09	1.21
Mathworks						1.0	99.0	99.0	1.0	1.0	99.0	1.0	1.07
Mentor Graphics							99.0	99.0	1.07	1.07	99.0	1.0	1.33
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.0	99.0	1.2	1.21
dSPACE											99.0	1.05	1.19
Elektrobit												99.0	99.0
Mecel													1.22
Etas	Incon: 0.00												

Figure A.41: Pair-wise comparisons for the criteria Service and Support of Model Tools

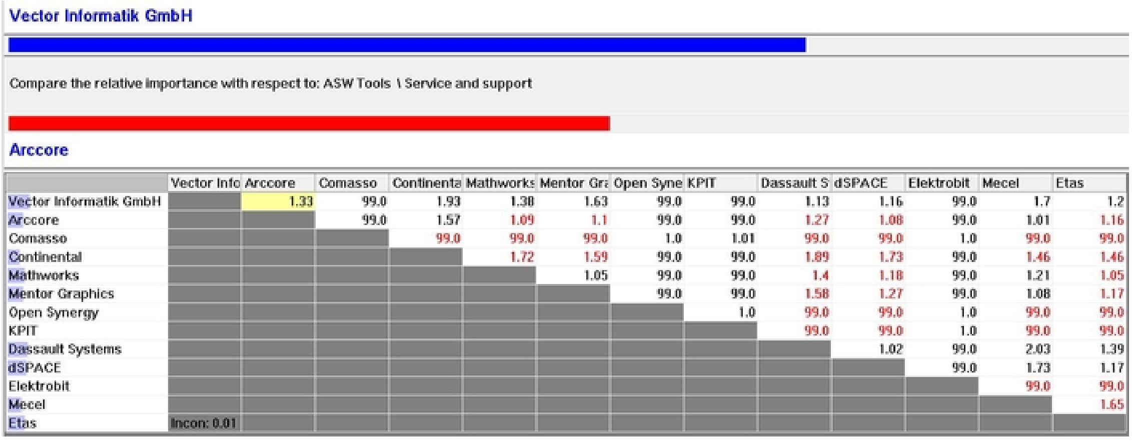


Synthesis with respect to: Service and support
(Goal: AUTOSAR TOOL CHAIN > Model Tools (L: .203) > Service and support (L: .)
Overall Inconsistency = .00

Vector Informatik GmbH	.118
Dassault Systems	.117
dSPACE	.115
Arccore	.113
Mathworks	.111
Etas	.111
Mentor Graphics	.107
Mecel	.106
Continental	.098
Comasso	.001
Open Synergy	.001
KPIT	.001
Elektrobit	.001

Figure A.42: Synthesis w.r.t. criteria - Service and Support of Model Tools

ASW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: ASW Tools \ Service and support

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.33	99.0	1.93	1.38	1.63	99.0	99.0	1.13	1.16	99.0	1.7	1.2
Arccore			99.0	1.57	1.09	1.1	99.0	99.0	1.27	1.08	99.0	1.01	1.16
Comasso				99.0	99.0	99.0	1.0	1.01	99.0	99.0	1.0	99.0	99.0
Continental					1.72	1.59	99.0	99.0	1.89	1.73	99.0	1.46	1.46
Mathworks						1.05	99.0	99.0	1.4	1.18	99.0	1.21	1.05
Mentor Graphics							99.0	99.0	1.58	1.27	99.0	1.08	1.17
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.02	99.0	2.03	1.39
dSPACE											99.0	1.73	1.17
Elektrobit												99.0	99.0
Mecel													1.65
Etas	Incon: 0.01												

Figure A.43: Pair-wise comparisons for the criteria Service and Support of ASW Tools

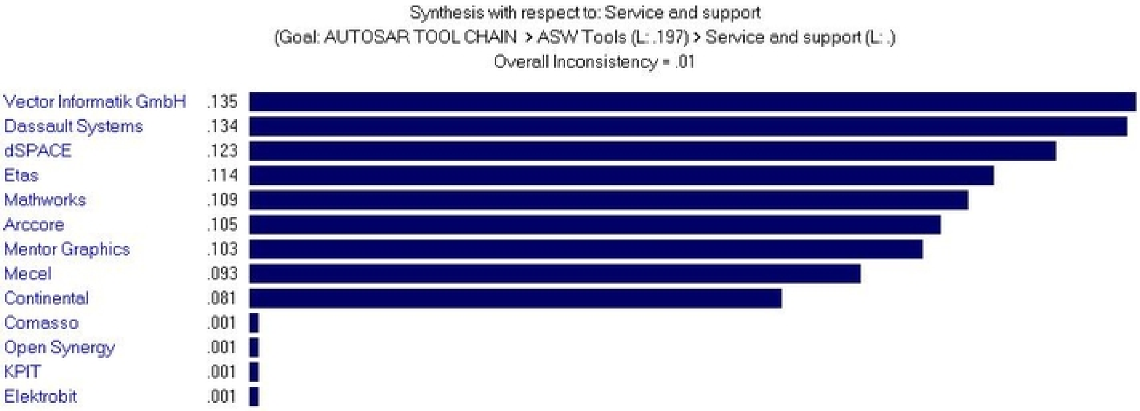
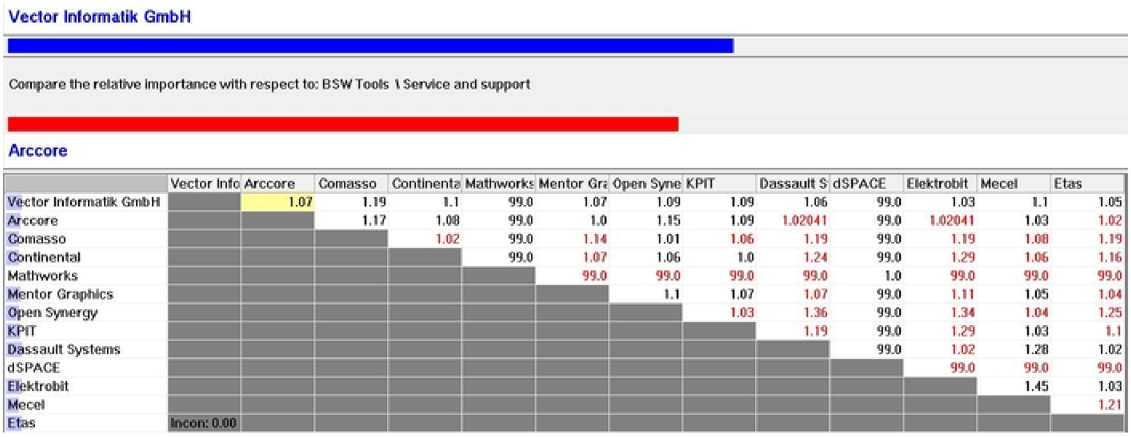


Figure A.44: Synthesis w.r.t. criteria - Service and Support of ASW Tools

BSW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: BSW Tools \ Service and support

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.07	1.19	1.1	99.0	1.07	1.09	1.09	1.06	99.0	1.03	1.1	1.05
Arccore			1.17	1.08	99.0	1.0	1.15	1.09	1.02041	99.0	1.02041	1.03	1.02
Comasso				1.02	99.0	1.14	1.01	1.06	1.19	99.0	1.19	1.08	1.19
Continental					99.0	1.07	1.06	1.0	1.24	99.0	1.29	1.06	1.16
Mathworks						99.0	99.0	99.0	99.0	1.0	99.0	99.0	99.0
Mentor Graphics							1.1	1.07	1.07	99.0	1.11	1.05	1.04
Open Synergy								1.03	1.36	99.0	1.34	1.04	1.25
KPIT									1.19	99.0	1.29	1.03	1.1
Dassault Systems										99.0	1.02	1.28	1.02
dSPACE											99.0	99.0	99.0
Elektrobit												1.45	1.03
Mecel													1.21
Etas	Incon: 0.00												

Figure A.45: Pair-wise comparisons for the criteria Service and Support of BSW Tools

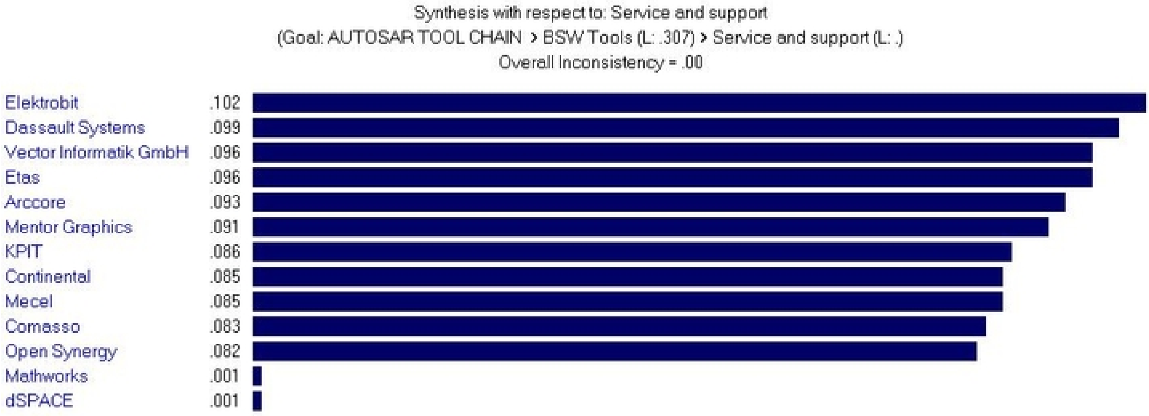
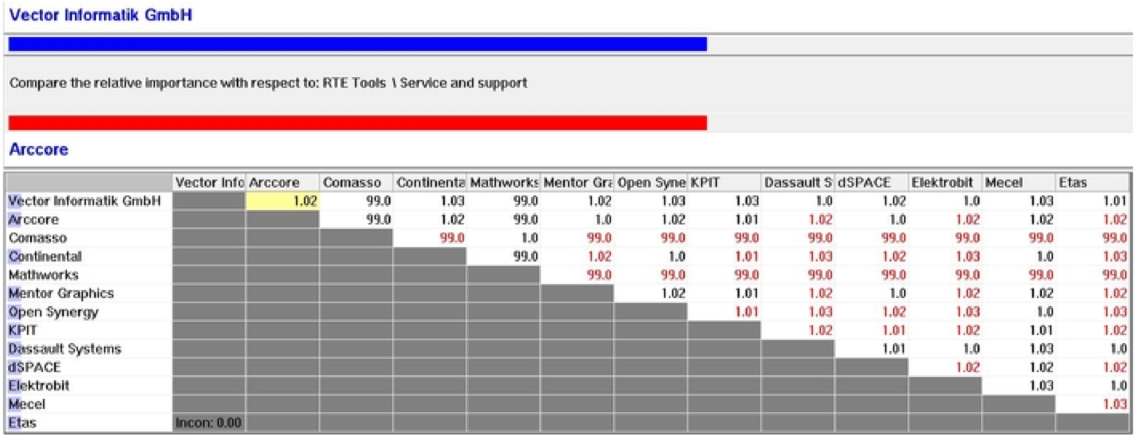


Figure A.46: Synthesis w.r.t. criteria - Service and Support of BSW Tools

RTE Tools



Vector Informatik GmbH

Compare the relative importance with respect to: RTE Tools \ Service and support

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.02	99.0	1.03	99.0	1.02	1.03	1.03	1.0	1.02	1.0	1.03	1.01
Arccore			99.0	1.02	99.0	1.0	1.02	1.01	1.02	1.0	1.02	1.02	1.02
Comasso				99.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Continental					99.0	1.02	1.0	1.01	1.03	1.02	1.03	1.0	1.03
Mathworks						99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Mentor Graphics							1.02	1.01	1.02	1.0	1.02	1.02	1.02
Open Synergy								1.01	1.03	1.02	1.03	1.0	1.03
KPIT									1.02	1.01	1.02	1.01	1.02
Dassault Systems										1.01	1.0	1.03	1.0
dSPACE											1.02	1.02	1.02
Elektrobit												1.03	1.0
Mecel													1.03
Etas	Incon: 0.00												

Figure A.47: Pair-wise comparisons for the criteria Service and Support of RTE Tools

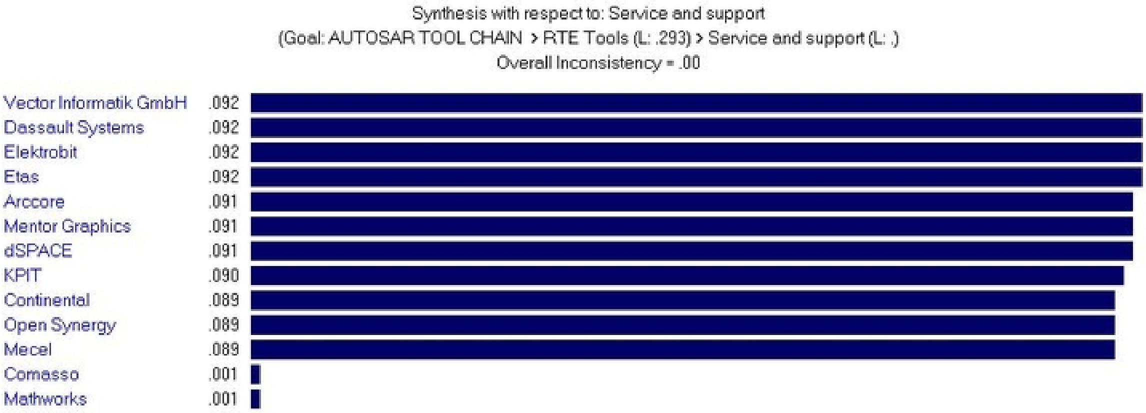
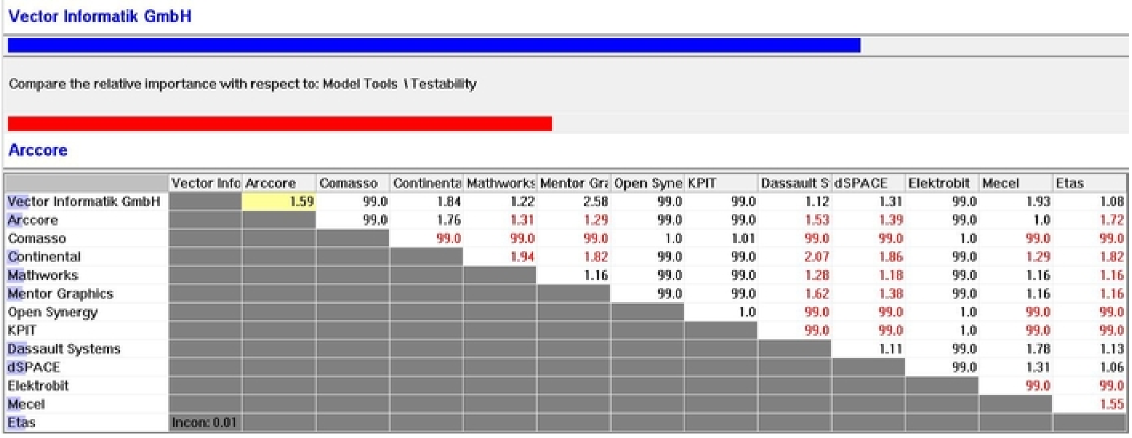


Figure A.48: Synthesis w.r.t. criteria - Service and Support of RTE Tools

A.2.7 Criteria - Testability

Model Tools



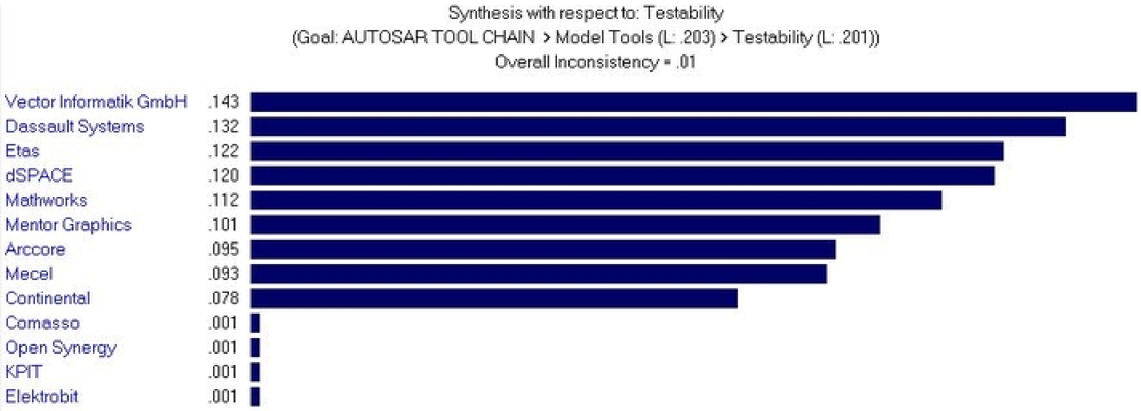
Vector Informatik GmbH

Compare the relative importance with respect to: Model Tools \ Testability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.59	99.0	1.84	1.22	2.58	99.0	99.0	1.12	1.31	99.0	1.93	1.08
Arccore			99.0	1.76	1.31	1.29	99.0	99.0	1.53	1.39	99.0	1.0	1.72
Comasso				99.0	99.0	99.0	1.0	1.01	99.0	99.0	1.0	99.0	99.0
Continental					1.94	1.82	99.0	99.0	2.07	1.86	99.0	1.29	1.82
Mathworks						1.16	99.0	99.0	1.28	1.18	99.0	1.16	1.16
Mentor Graphics							99.0	99.0	1.62	1.38	99.0	1.16	1.16
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.11	99.0	1.78	1.13
dSPACE											99.0	1.31	1.06
Elektrobit												99.0	99.0
Mecel													1.55
Etas	Incon: 0.01												

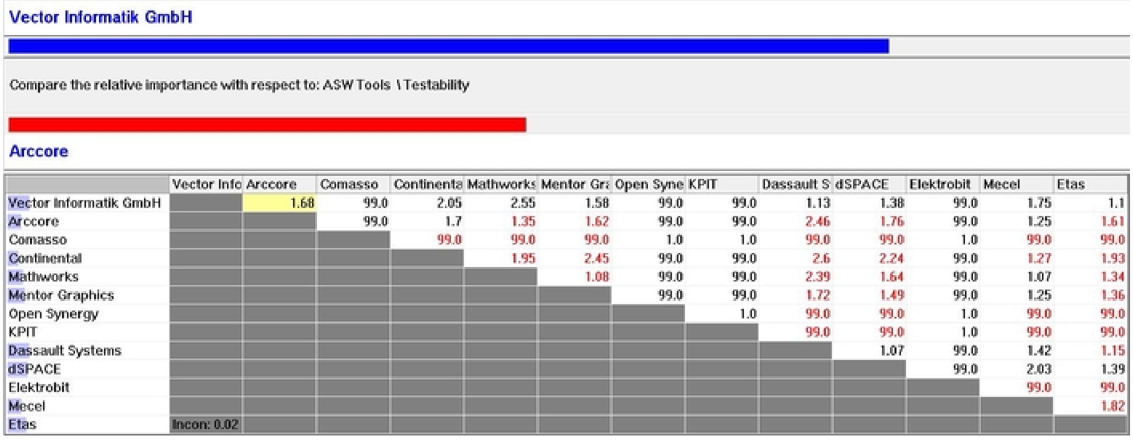
Figure A.49: Pair-wise comparisons for the criteria Testability of Model Tools



Synthesis with respect to: Testability
(Goal: AUTOSAR TOOL CHAIN > Model Tools (L: .203) > Testability (L: .201))
Overall Inconsistency = .01

Figure A.50: Synthesis w.r.t. criteria - Testability of Model Tools

ASW Tools



Vector Informatik GmbH

Compare the relative importance with respect to: ASW Tools \ Testability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gra	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.68	99.0	2.05	2.55	1.58	99.0	99.0	1.13	1.38	99.0	1.75	1.1
Arccore			99.0	1.7	1.35	1.62	99.0	99.0	2.46	1.76	99.0	1.25	1.61
Comasso				99.0	99.0	99.0	1.0	1.0	99.0	99.0	1.0	99.0	99.0
Continental					1.95	2.45	99.0	99.0	2.6	2.24	99.0	1.27	1.93
Mathworks						1.08	99.0	99.0	2.39	1.64	99.0	1.07	1.34
Mentor Graphics							99.0	99.0	1.72	1.49	99.0	1.25	1.36
Open Synergy								1.0	99.0	99.0	1.0	99.0	99.0
KPIT									99.0	99.0	1.0	99.0	99.0
Dassault Systems										1.07	99.0	1.42	1.15
dSPACE											99.0	2.03	1.39
Elektrobit												99.0	99.0
Mecel													1.82
Etas	Incon: 0.02												

Figure A.51: Pair-wise comparisons for the criteria Testability of ASW Tools

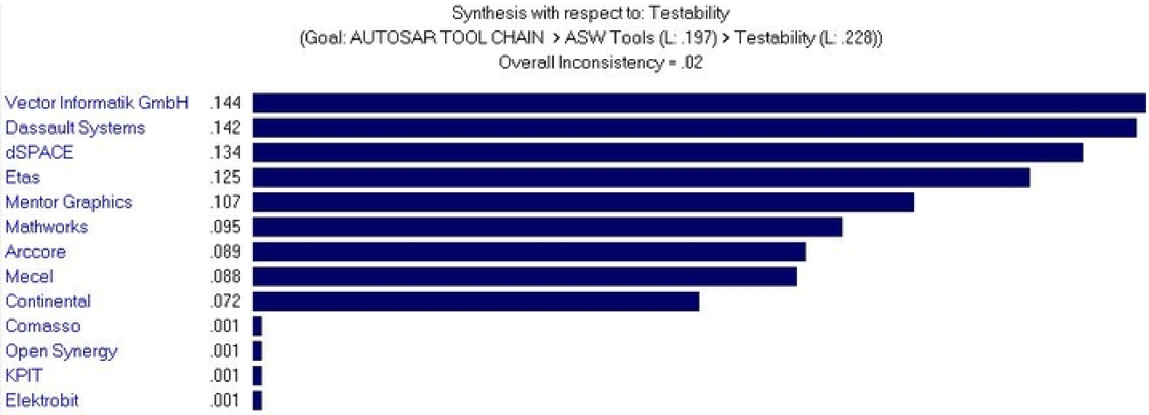
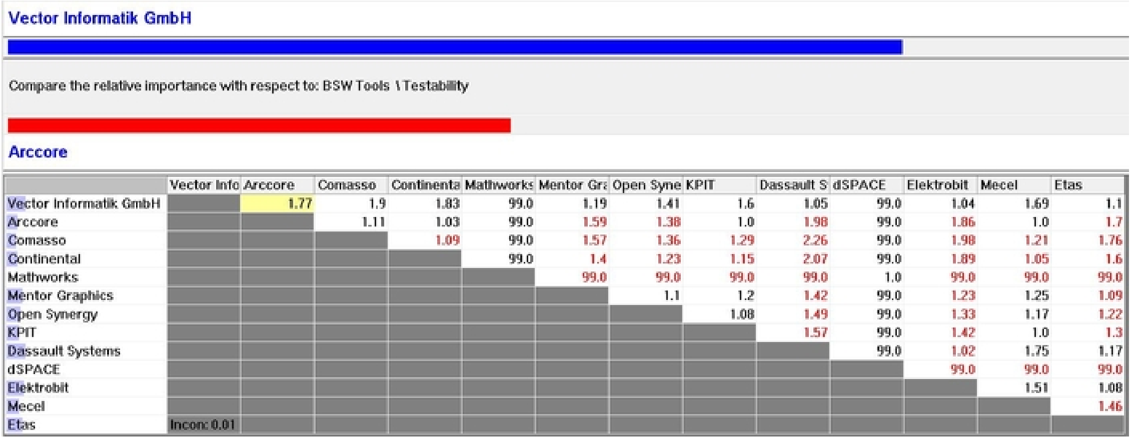


Figure A.52: Synthesis w.r.t. criteria - Testability of ASW Tools

BSW Tools



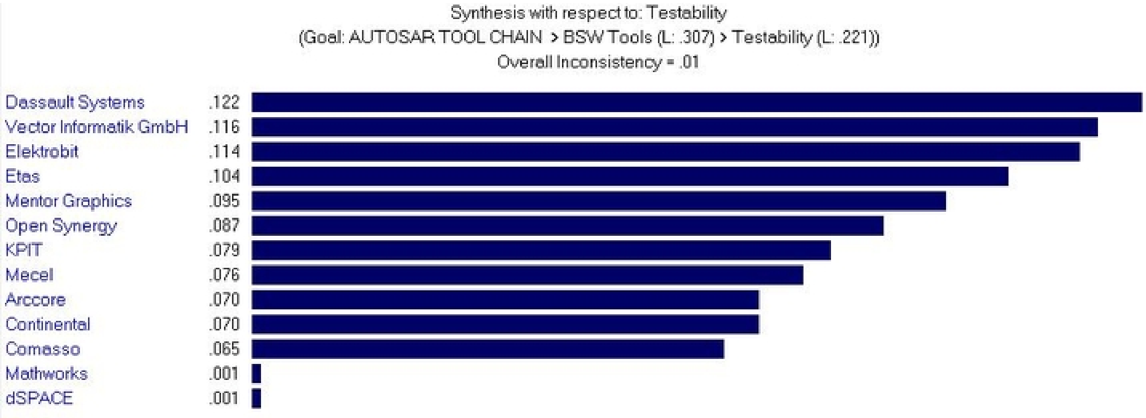
Vector Informatik GmbH

Compare the relative importance with respect to: BSW Tools \ Testability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.77	1.9	1.83	99.0	1.19	1.41	1.6	1.05	99.0	1.04	1.69	1.1
Arccore			1.11	1.03	99.0	1.59	1.38	1.0	1.98	99.0	1.86	1.0	1.7
Comasso				1.09	99.0	1.57	1.36	1.29	2.26	99.0	1.98	1.21	1.76
Continental					99.0	1.4	1.23	1.15	2.07	99.0	1.89	1.05	1.6
Mathworks						99.0	99.0	99.0	99.0	1.0	99.0	99.0	99.0
Mentor Graphics							1.1	1.2	1.42	99.0	1.23	1.25	1.09
Open Synergy								1.08	1.49	99.0	1.33	1.17	1.22
KPIT									1.57	99.0	1.42	1.0	1.3
Dassault Systems										99.0	1.02	1.75	1.17
dSPACE											99.0	99.0	99.0
Elektrobit												1.51	1.08
Mecel													1.46
Etas	Incon: 0.01												

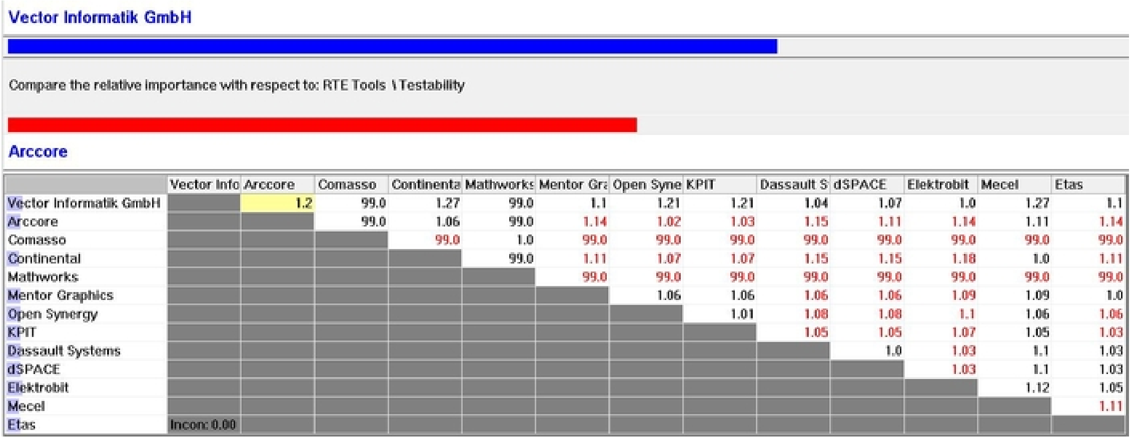
Figure A.53: Pair-wise comparisons for the criteria Testability of BSW Tools



Synthesis with respect to: Testability
(Goal: AUTOSAR TOOL CHAIN > BSW Tools (L: .307) > Testability (L: .221))
Overall Inconsistency = .01

Figure A.54: Synthesis w.r.t. criteria - Testability of BSW Tools

RTE Tools



Vector Informatik GmbH

Compare the relative importance with respect to: RTE Tools \ Testability

Arccore

	Vector Info	Arccore	Comasso	Continenta	Mathworks	Mentor Gr	Open Syne	KPIT	Dassault S	dSPACE	Elektrobit	Mecel	Etas
Vector Informatik GmbH		1.2	99.0	1.27	99.0	1.1	1.21	1.21	1.04	1.07	1.0	1.27	1.1
Arccore			99.0	1.06	99.0	1.14	1.02	1.03	1.15	1.11	1.14	1.11	1.14
Comasso				99.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Continental					99.0	1.11	1.07	1.07	1.15	1.15	1.18	1.0	1.11
Mathworks						99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
Mentor Graphics							1.06	1.06	1.06	1.06	1.09	1.09	1.0
Open Synergy								1.01	1.08	1.08	1.1	1.06	1.06
KPIT									1.05	1.05	1.07	1.05	1.03
Dassault Systems										1.0	1.03	1.1	1.03
dSPACE											1.03	1.1	1.03
Elektrobit												1.12	1.05
Mecel													1.11
Etas	Incon: 0.00												

Figure A.55: Pair-wise comparisons for the criteria Testability of RTE Tools

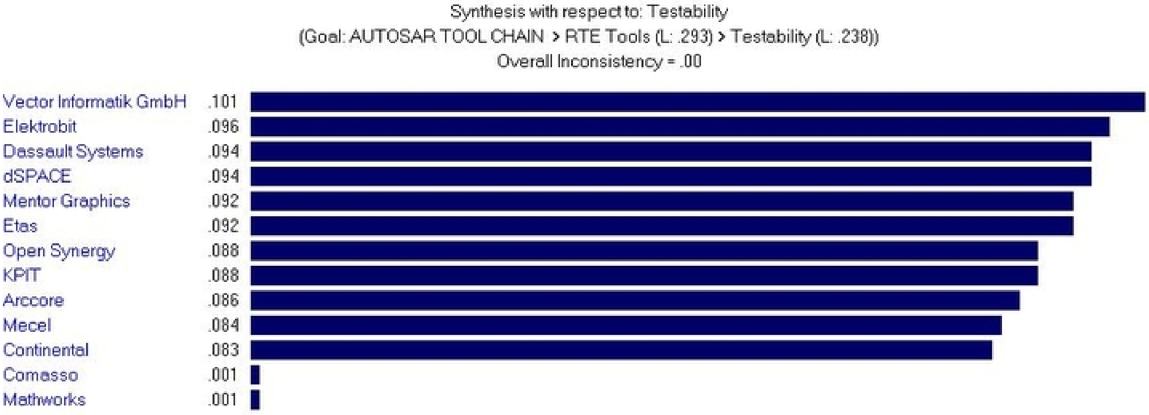
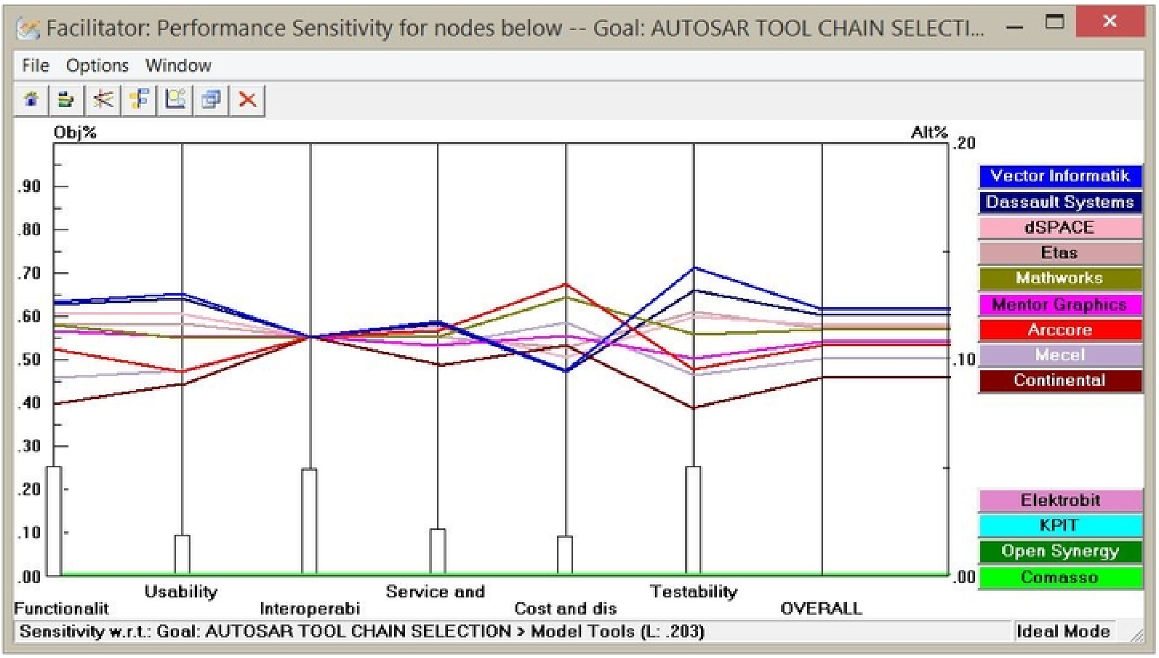


Figure A.56: Synthesis w.r.t. criteria - Testability of RTE Tools

A.3 Graphs - Based on Performance criteria

While assessing tools based on *performance*, high weights are placed on Functionality, Interoperability and Testability metrics when compared to other metrics like Usability, Service & Support and Cost & Distribution.

Model Tools



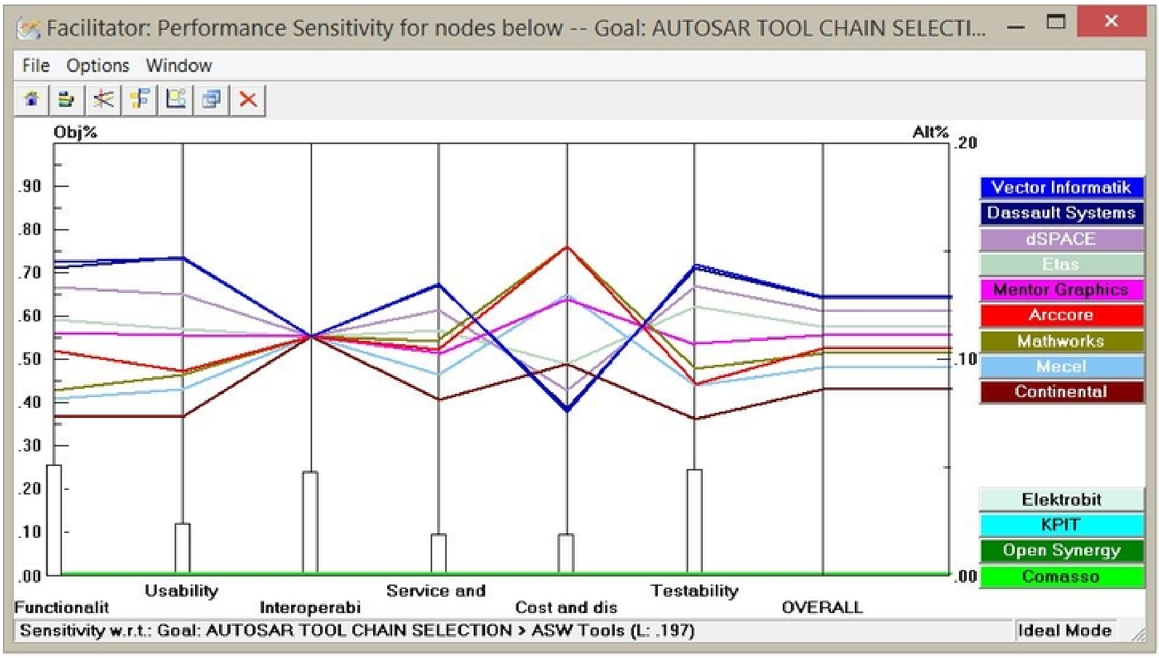
Facilitator: Performance Sensitivity for nodes below -- Goal: AUTOSAR TOOL CHAIN SELECTI...

Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > Model Tools (L: .203)

Ideal Mode

Figure A.57: Ranking of model tools prioritizing performance

ASW Tools



Facilitator: Performance Sensitivity for nodes below -- Goal: AUTOSAR TOOL CHAIN SELECTI...

Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > ASW Tools (L: .197)

Figure A.58: Ranking of ASW tools prioritizing performance

BSW Tools

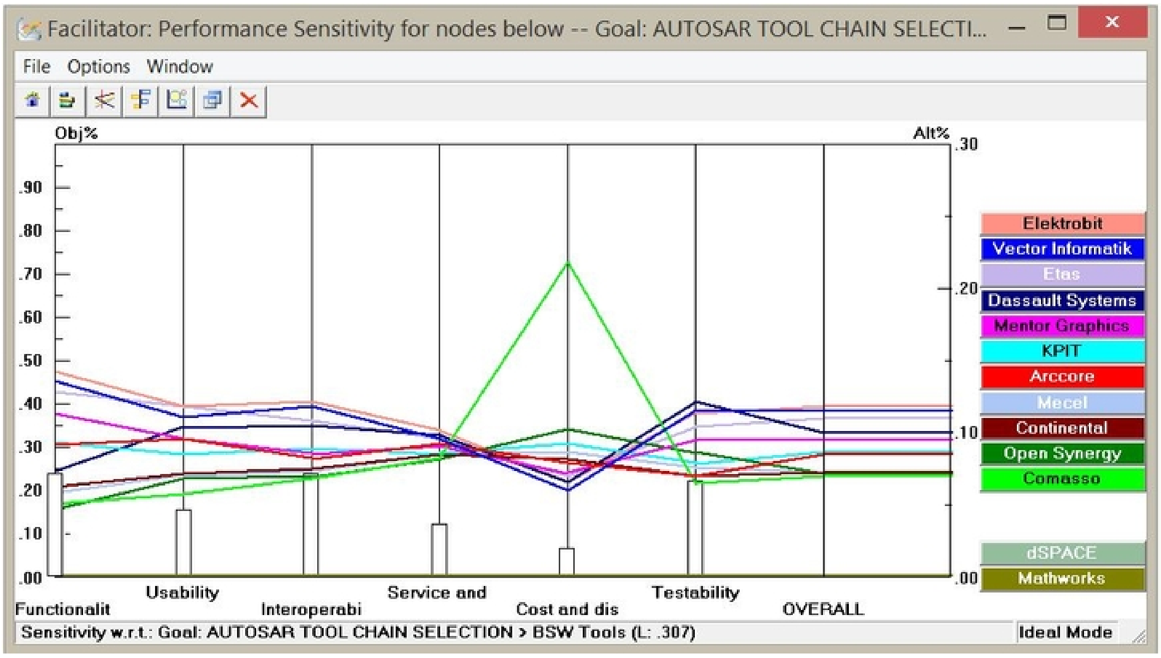
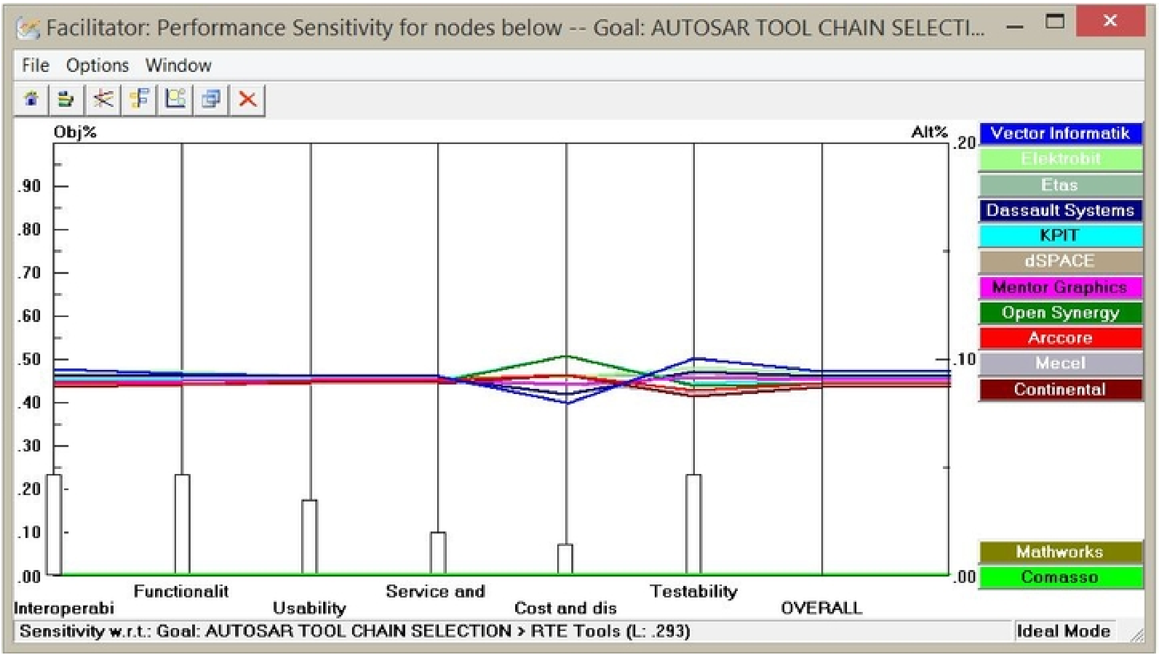


Figure A.59: Ranking of BSW tools prioritizing performance

RTE Tools


Facilitator: Performance Sensitivity for nodes below -- Goal: AUTOSAR TOOL CHAIN SELECTI...

Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > RTE Tools (L: .293)

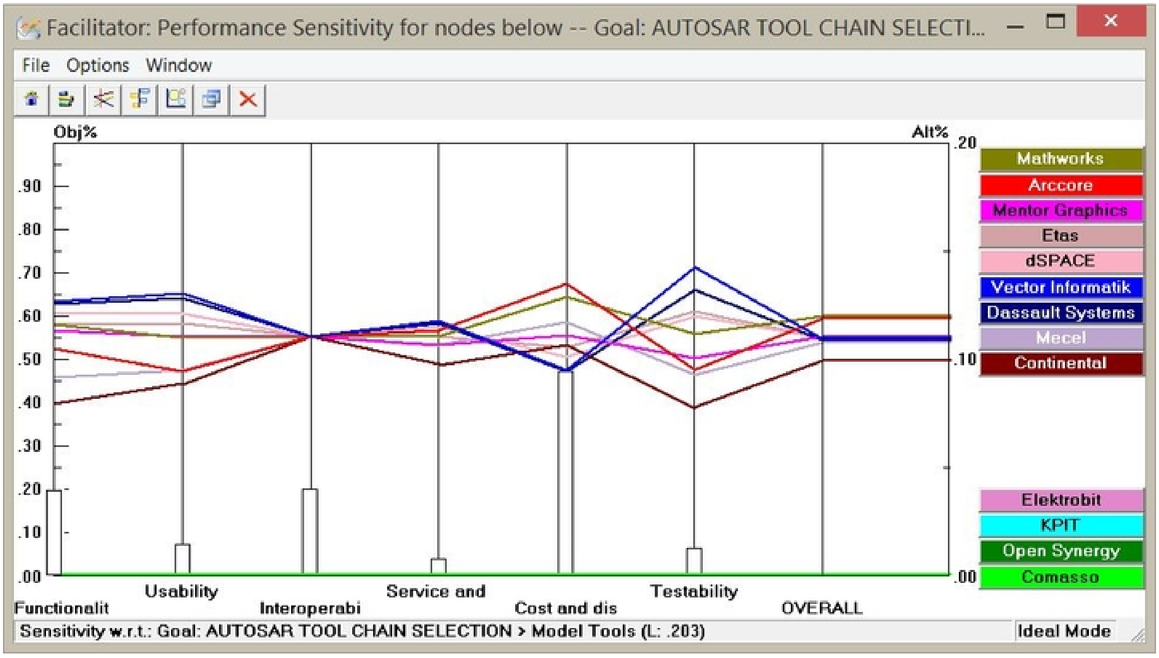
Ideal Mode

Figure A.60: Ranking of RTE tools prioritizing performance

A.4 Graphs - Based on Low-cost criteria

While assessing tools based on *low-cost* aspect, high emphasis is given to Cost & Distribution compared to all other criteria.

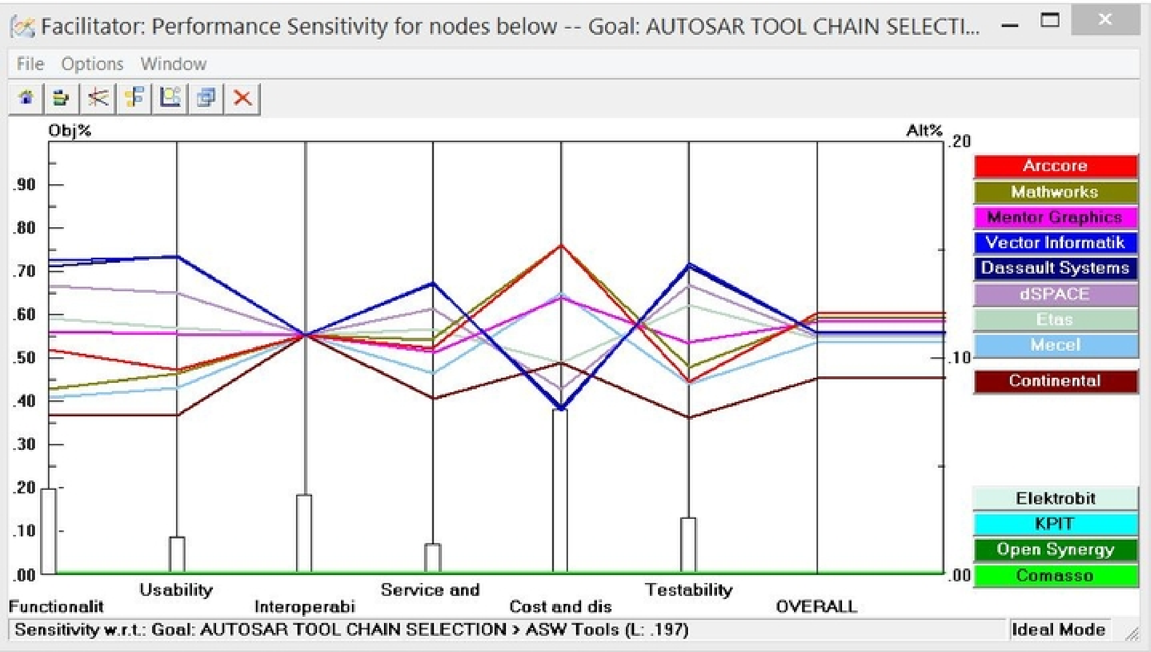
Model Tools



Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > Model Tools (L: .203)

Figure A.61: Ranking of model tools prioritizing cost

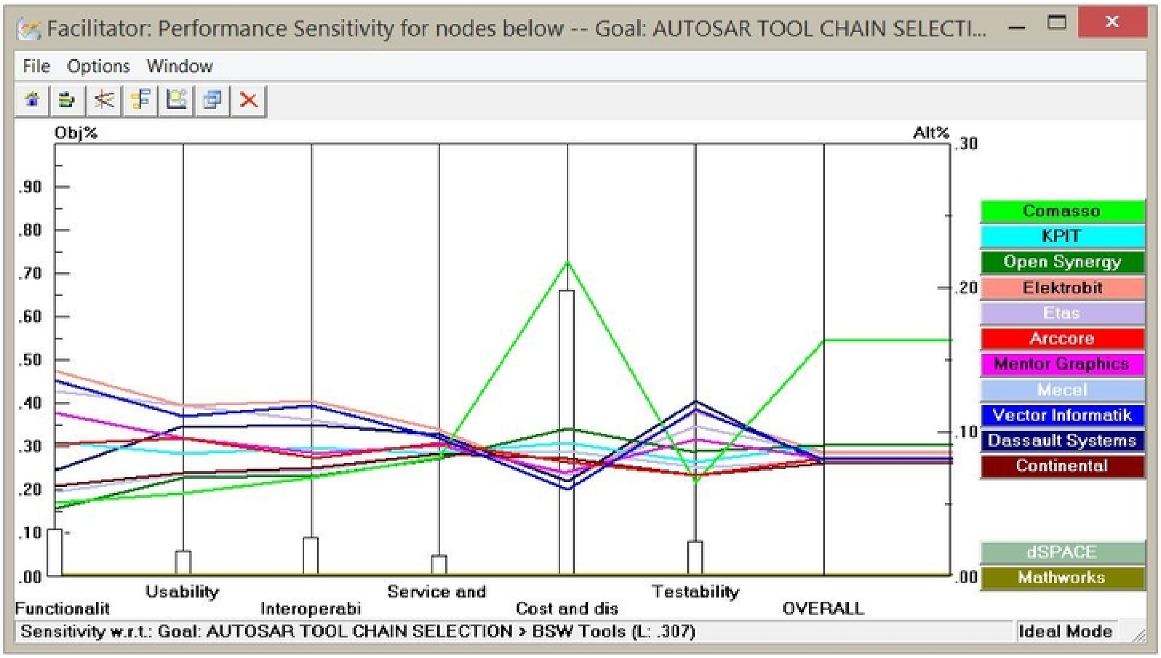
ASW Tools


Facilitator: Performance Sensitivity for nodes below -- Goal: AUTOSAR TOOL CHAIN SELECTI...

Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > ASW Tools (L: .197)

Figure A.62: Ranking of ASW prioritizing cost

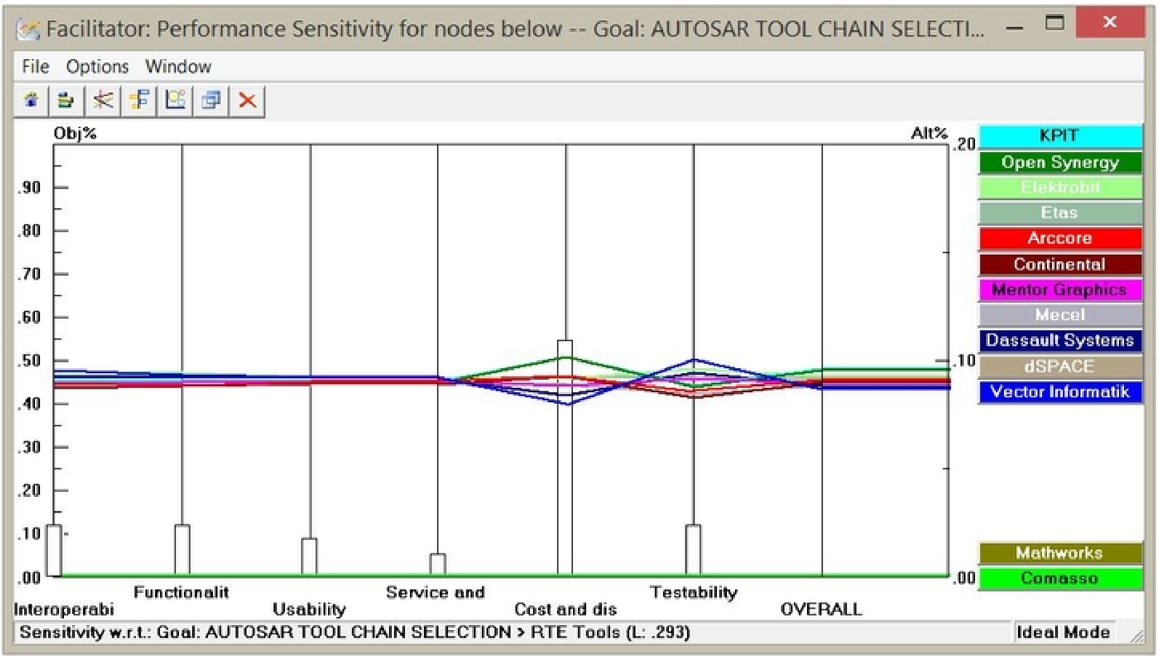
BSW Tools



Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > BSW Tools (L: .307)

Figure A.63: Ranking of BSW tools prioritizing cost

RTE Tools



Sensitivity w.r.t.: Goal: AUTOSAR TOOL CHAIN SELECTION > RTE Tools (L: .293)

Figure A.64: Ranking of RTE tools prioritizing cost

Table A.1: AUTOSAR tool cost analysis

Tool vendors	**Support for MCAL layer modules**	**Support for BSW code generation / configuration**	**Support for RTE generation**	**Support for ASW code generation**	**Support for AUTOSAR modeling**
ArcCore	🟩	🟩	🟩	■	■
Comasso	■	■	□	□	□
Continental	🟩	🟩	🟩	🟩	🟩
Dassault Systems	□	🟥	🟥	🟥	🟥
dSPACE	□	□	🟥	🟥	🟥
Elektrobit	🟥	🟥	🟩	□	□
ETAS	🟥	🟥	🟩	🟩	🟩
KPIT	🟩	🟩	🟩	□	□
Mathworks	□	□	□	■	■
Mecel	🟩	🟩	🟩	🟩	🟩
Mentor Graphics	🟩	🟩	🟩	🟩	🟩
Opensynergy	🟩	🟩	🟩	□	□
Vector Informatik Gmbh	🟥	🟥	🟥	🟥	🟥

■	Economical	🟩	Intermediate	🟥	Expensive	□	Not supported